MASTER EN SCIENCES INFORMATIQUES

Évaluation et amélioration d'un logiciel de gestion des comptes pour personnes handicapées mentales

Berger, Olivier; Cheron, France

Award date:
1990

Awarding institution:
Universite de Namur

Année académique 1989 - 1990

Evaluation et amélioration d'un logiciel de gestion des comptes pour personnes handicapées mentales.

France CHERON **Olivier BERGER**

Promoteur : Madame Monique Noirhomme

Co-promoteur : Monsieur Michel Mercier, professeur au Département de psychologie, F.N.D.P.

Mémoire présenté en vue de l'obtention du diplôme de licence et maîtrise en informatique.

ABSTRACT

Ce travail présente l'évaluation d'un logiciel de tenue de comptes pour personnes handicapées mentales. Celle-ci se base sur une démarche originale et logique. Les résultats des expérimentations ont donné lieu, d'une part, à des modifications et à des extensions apportées au logiciel et, d'autre part, au développement de nouveaux instruments d'appréciation et à la conception d'une nouvelle modalité d'expérimentation.

This work exposes the evaluation of a program of accounting for mentally disabled persons. This one is based on an original and logical proceeding. The results of the experiments gave way, on one side, to changes and extents brought to the software, on the other side, to the development of new appreciation tools and to the establishment of a new experiment modality.

Nous remercions chaleureusement Madame Noirhomme et Monsieur Mercier qui ont accepté de parrainer ce travail et nous ont prodigué leurs encouragements et leurs conseils.

Nous témoignons également notre reconnaissance à Monsieur Baechler pour les conseils judicieux qu'il nous a donnés lors de notre séjour en Suisse.

Nous adressons également nos remerciements à Gilles Le Cardinal, Martine Galand, Anne Faton, Jean-Pierre Peters et Vincent Mineur.

Nous tenons également à manifester notre reconnaissance aux éducateurs qui ont su nous accorder de longues heures de travail et de discussions.

Enfin, nous exprimons notre gratitude à Séverine et Eric sans qui ce travail n'aurait pu être terminé dans les délais, à Jean-François, Bertrand et Stéphane pour leur indispensable soutien ainsi qu'à toutes les personnes qui, de façons très diverses, nous ont apporté leur aide durant la réalisation de ce travail.

"Des deux partenaires de l'aide, d'habitude, seul l'aidant parle; l'autre, le plus souvent, se tait. Sur l'aide pourtant, la personne aidée garde le dernier mot..." (R.P. Xavier Dijon)

TABLE DES MATIERES

INTRODUCTION

Notre travail se situe à l'intersection entre deux pôles de recherche: le domaine de l'informatique et du handicap mental et celui de l'évaluation de logiciels.

Le domaine de l'informatique et du handicap, encore peu exploré il y a quelques années, semble aujourd'hui connaître de grands développements. Une littérature et de brillantes réalisations commencent à apparaître. Ce domaine doit encore être considéré à l'heure actuelle comme relevant de la recherche. Néanmoins, la plupart de ces réalisations ont pour objet l'aide à des personnes atteintes d'un handicap physique. Des solutions informatiques, le plus souvent de type "hardware", sont utilisées comme prothèses. De jour en jour, grâce aux progrès technologiques tels que la synthèse vocale, la reconnaissance de la parole et des formes,... qui émanent des chercheurs en intelligence artificielle et des ingénieurs, il est possible d'accomplir des exploits de plus en plus admirables.

Nous pouvons cependant déplorer le manque de projets informatiques adressés à des personnes atteintes d'un handicap mental. Les études théoriques à ce sujet sont également fort rares. Face à l'absence de logiciels, les personnes handicapées mentales sont souvent amenées à utiliser des programmes adressés à des enfants. Mais ces deux types de populations n'ont pas les mêmes besoins ni les mêmes capacités. Des personnes handicapées mentales ont un vécu et une histoire beaucoup plus importants que ceux des enfants. Ils ont des besoins spécifiques tels que celui d'obtenir une aide pour gérer leur budget. Par ailleurs, la capacité et la vitesse d'apprentissage sont plus faibles pour des personnes handicapées mentales que pour des enfants. Dès lors, la plupart des logiciels éducatifs disponibles sur le marché et destinés aux enfants sont souvent inadaptés à la population handicapée.

Si les projets informatiques sont peu nombreux en ce domaine, un véritable besoin existe. Nous sommes convaincus qu'une aide efficace et

pertinente peut être apportée aux handicapés mentaux grâce au développement de logiciels conçus pour eux. Le développement devra se faire le plus souvent possible avec leur collaboration et toujours en vue de leur offrir un produit qui réponde à leurs besoins en tenant compte de leurs capacités et de leurs particularités. Puisque la logique de ces personnes et la nôtre sont différentes, il convient toujours de vérifier si l'outil réalisé à leur intention leur apporte véritablement une aide.

L'évaluation de logiciels est reconnue par la plupart des auteurs comme une phase essentielle du cycle de vie de tout projet informatique. Néanmoins, ce domaine ne semble pas particulièrement inspirer les chercheurs... Nous n'avons en effet trouvé que très peu de travaux à ce propos.

Le sujet de notre mémoire se trouve à l'intersection entre ces deux domaines et est, à notre connaissance, pour ainsi dire inexploré. Notre travail ne repose donc pas sur de nombreux éléments théoriques mais plutôt sur ce que le bon sens et la logique nous ont dicté. Voyons à présent le fil conducteur que nous avons suivi au cours de notre travail.

Dans le **chapitre 1**, nous précisons ce que nous entendons par l'évaluation d'un logiciel, nous voyons les buts poursuivis par une évaluation en la situant dans le cycle de vie d'un projet informatique, nous abordons ensuite les difficultés et particularités liées à l'évaluation d'un projet destiné à des personnes handicapées mentales. Enfin, après avoir relevé les acteurs impliqués dans l'évaluation et leurs rôles respectifs, nous présentons de manière assez formelle la démarche que nous avons imaginée afin d'évaluer le logiciel Comptes.

Le **chapitre 2** a pour but de présenter brièvement le logiciel sur lequel porte notre évaluation. Il s'agit d'un programme de tenue de comptes qui a été développé lors de l'année académique 1988-1989 par Marc Déplechin et Gianni Strappazzon, deux "mémorants en informatique". Ce logiciel s'adresse à des personnes handicapées mentales. Nous évoquons la

naissance du projet, la logique, les fonctionnalités, les interfaces ainsi que les particularités de ce programme.

Le **chapitre 3** concerne l'évaluation proprement dite de ce logiciel. Celle-ci s'appuie sur la démarche exposée dans le premier chapitre. Nous commençons par relever un ensemble d'objectifs poursuivis par l'évaluation, nous exposons les conditions d'expérimentation utilisées pour répondre aux objectifs, nous examinons ensuite les différents outils d'appréciation employés lors de l'expérimentation. Pour chacun d'eux, nous montrons leurs intérêts par rapport aux objectifs et la façon dont ils peuvent être exploités afin de tirer des conclusions. Après avoir exposé les problèmes que nous avons rencontrés lors des expérimentations, nous présentons, pour chaque objectif d'évaluation, les résultats auxquels nous sommes arrivés. En outre, nous critiquons les différents outils d'appréciation utilisés.

Le **chapitre 4** présente au lecteur la seconde version du logiciel que nous avons développée. L'évaluation du logiciel montre en effet qu'un certain nombre d'extensions fonctionnelles et d'améliorations ergonomiques s'imposent.

Afin de recueillir des résultats d'évaluation les plus fins et les plus précis possibles, il est impératif de pouvoir disposer d'un nombre élevé d'outils d'appréciation différents. En effet, les conclusions sur les différents outils d'appréciation ont mis en évidence les avantages et les inconvénients de chacun d'eux et ont montré la nécessité d'opérer à des croisements entre outils. Le **chapitre 5** présente donc les améliorations apportées aux outils d'appréciation existants, expose les nouveaux outils imaginés et développés et présente au lecteur les objectifs et les fonctionnalités d'un programme d'aide à l'évaluation que nous avons réalisé.

Pour terminer cette nouvelle boucle dans le cycle de vie du projet, nous exposons naturellement l'évaluation de cette seconde version du programme dans le **chapitre 6** en illustrant l'utilisation de nouveaux outils de mesure.

Dans le **chapitre 7**, nous exposons une nouvelle modalité d'expérimentation que nous avons imaginée suite aux problèmes rencontrés dans les phases d'évaluation. Nous montrons également l'application de

nouveaux instruments d'évaluation à ces nouvelles conditions et leur exploitation par les mathématiques floues.

Enfin, dans le **chapitre 8**, nous tirons quelques conclusions sur l'ensemble de ce travail et nous présentons des prolongements possibles à ce mémoire.

CHAPITRE I : L'EVALUATION DE LOGICIELS

Avant d'évaluer le logiciel Comptes, il convient d'exposer ce que nous entendons par l'évaluation d'un logiciel, de préciser les buts poursuivis par une telle évaluation, de montrer les difficultés liées à l'évaluation d'un logiciel pour personnes handicapées mentales et enfin d'exposer une démarche cohérente pour réaliser cette évaluation.

1.1. DEFINITIONS

1.1.1. L'EVALUATION

Evaluer, c'est:
- déterminer (une quantité) par le calcul sans recourir à la mesure directe;
- fixer approximativement (apprécier, estimer, juger).

Cette définition du Petit Robert met en évidence une des difficultés de notre travail. Pour évaluer, nous ne pouvons que très rarement disposer de mesures. Ce n'est que sur base d'appréciations que nous pouvons construire l'évaluation.

1.1.2. L'EVALUATION DE LOGICIELS

Remarquons que, dans le cadre de ce mémoire, nous n'entendons pas traiter l'évaluation de logiciels sous l'aspect de mesure de performances ni sous l'aspect financier.

Pour nous, évaluer un logiciel, c'est apprécier si le logiciel répond bien à ses spécifications et aux objectifs qui lui ont été conférés au moment de la rédaction du cahier de charges, nous évaluons donc pour rendre compte des écarts éventuels entre les objectifs assignés au projet et les résultats obtenus lors de l'utilisation du programme.

1.2. LES BUTS POURSUIVIS PAR L'EVALUATION

La mise en évidence de ces écarts permettra de décider de l'avenir du projet. L'évaluation est une phase essentielle du cycle de vie de tout projet informatique car c'est elle qui validera ou, au contraire, remettra en question les options qui ont été prises au cours des étapes antérieures. Ainsi, les choix concernant, par exemple, les spécifications, la conception, la réalisation et le matériel seront critiqués lors de cette étape.
L'évaluation permettra ainsi:

- d'abandonner purement et simplement le projet si les écarts entre les objectifs assignés au projet et les observations récoltées sont jugés trop importants et si on ne peut trouver aucun autre choix qui permettrait de répondre aux objectifs. Le projet semble donc voué à l'échec;

- de revoir les choix posés au début du cycle de vie du projet si, malgré des écarts jugés trop importants, on peut envisager des alternatives qui ont des chances de réussir. Ces autres choix peuvent s'inspirer des résultats de l'évaluation;

- d'améliorer et d'étendre le logiciel afin qu'il réalise mieux les objectifs qui lui ont été assignés. On se trouve dans le cas où les écarts ne sont pas trop importants mais où il est possible et utile de les diminuer. Il s'agira par exemple de :
. supprimer des fonctionnalités inutiles et inutilisées;
. ajouter des fonctionnalités de manière à étendre la portée du logiciel;
. améliorer l'ergonomie c'est-à-dire développer de nouvelles interfaces, envisager d'autres outils d'interaction, d'autres styles d'interaction,...

- de mettre un terme au projet. Si les écarts relevés sont faibles ou inférieurs à un seuil (qu'il convient de fixer), on peut conclure à la réussite du projet. La distribution et l'utilisation du logiciel peuvent alors être entreprises.

1.3. DIFFICULTES ET PARTICULARITES D E L'EVALUATION D'UN LOGICIEL POUR PERSONNES HANDICAPEES MENTALES

Habituellement, lors de l'évaluation d'un logiciel, c'est l'utilisateur lui-même qui est et doit être l'interlocuteur de l'informaticien. En effet, il est le mieux placé pour juger de l'(in)adéquation entre les objectifs du programme et ses réalisations. Il peut notamment émettre toute une série de critiques concernant l'interface et les fonctionnalités offertes par le logiciel.

A notre avis, les difficultés liées à l'évaluation d'un logiciel pour personnes handicapées mentales sont au nombre de trois.

Premièrement, la logique et la rationnalité de ces utilisateurs sont particulières et ne correspondent pas nécessairement aux nôtres, elles peuvent nous échapper. Or, il faut respecter leur logique puisque le programme leur est destiné. Dès lors, la difficulté rencontrée par les différents acteurs impliqués dans le processus d'évaluation consiste à essayer, le plus souvent possible, de faire abstraction de leur propre logique au profit de celle des utilisateurs.

En outre, la population handicapée est fortement hétérogène. S'il est vrai que deux utilisateurs ne réagissent jamais de la même façon, ce phénomène est d'autant plus important avec des utilisateurs handicapés mentaux. Toutes ces personnes réagissent différemment et sont fort distinctes. De plus, une même personne handicapée peut avoir des réactions fort différentes suivant les moments où on l'observe.

Enfin, les utilisateurs handicapés mentaux ont souvent des difficultés de communication et ne sont pas toujours aptes à exprimer aux informaticiens leurs difficultés, leurs critiques, leurs motivations, leurs intérêts pour le projet. De plus, certaines personnes pourraient peut-être avoir quelques difficultés à poser un jugement sur une situation.

De plus, s'il nous semble clair que, dans l'évaluation de tout projet informatique, une expérimentation avec les utilisateurs est utile, lorsqu'il s'agit de logiciels pour personnes handicapées mentales, au vu de ces difficultés, une expérimentation se révèle indispensable. Il faut recourir à l'aide d'autres acteurs qui, grâce à leur connaissance des personnes handicapées mentales, pourront en être les porte-paroles auprès des informaticiens et des responsables de l'évaluation. Ce sont les éducateurs et les psychologues.

Nous n'avons trouvé que très peu de réflexions dans la littérature au sujet de l'évaluation de logiciels et nous n'avons trouvé aucun travail concernant l'évaluation de logiciels pour personnes handicapées mentales. Afin d'évaluer concrètement un logiciel, nous avons dû concevoir une méthode de travail cohérente.

1.4. LA DEMARCHE SUIVIE POUR L'EVALUATION

Voici la démarche que nous avons mise sur pied et que nous avons suivie lors de notre évaluation.

1) Fixer les objectifs de l'évaluation

Pour mener à bien une évaluation pertinente, il convient de fixer une série d'objectifs. Ces objectifs de l'évaluation sont, pour la plupart, calqués sur ceux du logiciel. Mais, par ailleurs, on peut aussi s'intéresser à d'autres éléments tels que la durée de la période d'apprentissage, la qualité de l'ergonomie et la cohérence des prérequis.

2) Fixer les conditions dans lesquelles se dérouleront les expérimentations

Ces conditions d'expérimentation sont définies en fonction des objectifs poursuivis par l'évaluation, de la qualité des résultats que l'on veut obtenir et de la disponibilité des acteurs impliqués dans l'évaluation du logiciel à savoir les informaticiens, les éducateurs, les psychologues, les statisticiens et les utilisateurs.

3) Définir les moyens d'action de l'évaluation

Ces moyens d'action sont des instruments qui permettent de recueillir de l'information concernant les interactions entre les utilisateurs et le logiciel. Il faut toujours s'assurer de disposer d'un ou de plusieurs outils d'appréciation pour pouvoir répondre à chaque objectif de l'évaluation.

4) Réaliser l'expérimentation

L'expérimentation débute par l'introduction du logiciel auprès des utilisateurs. Au cours de l'expérimentation, il est nécessaire de s'assurer que les conditions fixées sont respectées dans chaque site d'expérimentation et qu'une série d'appréciations est ainsi récoltée grâce aux outils.

5) Tirer les résultats de l'évaluation

On récupère les appréciations obtenues grâce aux outils au cours de l'expérimentation. Il est important de les mettre en regard avec les objectifs de l'évaluation de manière à dégager les écarts qui sont les résultats de l'évaluation. Ceux-ci permettent de décider de l'avenir du projet (Cfr. 1.2.). Le statisticien est ici d'une grande aide dans le traitement des appréciations.

REMARQUES:

Les personnes qui mènent une évaluation ne sont pas neutres et objectives. Elles doivent bien entendu essayer d'intervenir le moins possible dans l'évaluation. Mais elles se font inévitablement une opinion du logiciel qu'elles évaluent et, au cours des différentes étapes de la démarche que nous proposons, elles posent des choix qui peuvent avoir des conséquences sur l'ensemble de l'évaluation. Elles fixent, par exemple, les objectifs de l'évaluation en fonction de leurs connaissances, de leur sensibilité. D'autres personnes pourraient déterminer d'autres objectifs. De même, elles choisissent et définissent les outils de l'évaluation, elles sélectionnent les sites d'expérimentation, les méthodes d'exploitation des appréciations obtenues,... Bien sûr, malgré toutes ces sources de subjectivité, l'évaluateur doit tenter à tout moment de rester le plus neutre possible.

Différents types de spécialistes interviennent au cours de l'évaluation d'un logiciel pour personnes handicapées mentales. Nous avons vu en effet que plusieurs acteurs doivent prendre part à l'évaluation: l'informaticien, le psychologue, l'éducateur, le statisticien et bien sûr l'utilisateur. Une approche pluridisciplinaire s'impose.

Cette démarche ne prétend pas être une méthodologie rigoureuse à suivre pour l'évaluation de tout projet pour personnes handicapées mentales. Elle montre de manière un peu formelle la manière dont nous avons réalisé et entendu l'évaluation du programme Comptes. Nous espérons seulement qu'elle puisse inspirer d'autres personnes chargées de l'évaluation d'un projet informatique destiné à des personnes handicapées mentales.

CHAPITRE II : PRESENTATION DE LA PREMIERE VERSION DU LOGICIEL DE TENUE DE COMPTES

Après avoir décrit de manière assez formelle l'évaluation, nous présentons dans ce chapitre le logiciel sur lequel doit porter notre évaluation. Il est le fruit du travail réalisé par Marc Déplechin et Gianni Strappazzon. Nous exposons ici, le plus fidèlement possible, un résumé de leur travail. Le lecteur intéressé peut se référer à leur mémoire "Un logiciel de gestion des comptes pour personnes handicapées mentales: développement et évaluation".

2.1. LA NAISSANCE DU PROJET

Le projet d'un "logiciel de gestion de budget pour personnes handicapées mentales" répond à une demande d'institutions belges pour personnes handicapées mentales qui tentent différentes expériences de formation à la gestion du budget. Le but initial du logiciel est d'aider des personnes handicapées mentales adultes, relativement autonomes, à gérer leur budget hebdomadaire ou mensuel.

Ces institutions ont fait appel au centre PSINHA (PSychologie - INformatique - HAndicap), un service de recherche, de documentation et d'information qui traite de problèmes liés à l'informatique et au handicap, et qui dépend du Département de Psychologie de la Faculté de Médecine de Namur. Celui-ci lance le projet en créant un groupe de travail. Ce groupe établit un cahier de charges qui donne lieu à une première version du logiciel réalisée sur PC en Turbo Pascal. Cependant, cette version n'est pas jugée satisfaisante par le groupe de travail, notamment à cause d'un graphisme peu développé. Dès lors, la décision est prise de développer le logiciel sur Commodore Amiga. En effet, d'une part, cette machine offre

de nombreuses capacités sonores et graphiques fort intéressantes pour une application interactive avec des personnes handicapées mentales, d'autre part, son prix est très abordable pour les institutions susceptibles de l'acquérir.

Dès ce moment, la Fondation Suisse pour les Téléthèses (FST), ainsi que plusieurs institutions suisses pour personnes handicapées mentales, qui disposent déjà de Commodore Amiga, prennent part également au projet. Une nouvelle réflexion, basée sur le logiciel existant, est alors entamée dans le cadre du mémoire de Marc Déplechin et Gianni Strappazzon en collaboration étroite avec Monsieur Baechler qui fait partie de la FST et d'une institution suisse, La Castalie.

Au cours de cette réflexion, les objectifs du logiciel de gestion de budget sont élargis. Il s'agit encore de fournir un outil d'autonomie permettant à certains handicapés de gérer réellement leur budget, mais le logiciel peut aussi être considéré comme un outil d'apprentissage permettant à la personne handicapée mentale de mieux comprendre des concepts et termes liés à la gestion de budget tels que recette, dépense, porte-monnaie, équilibre, tenue de comptes,... et d'acquérir de nouvelles habitudes et une meilleure notion de la valeur de l'argent.

2.2. LA POPULATION CIBLE

Le logiciel de gestion de budget s'adresse à des personnes handicapées mentales qui reçoivent une certaine somme d'argent dont elles peuvent disposer librement. Ces personnes peuvent présenter de légers handicaps physiques. Néanmoins, le logiciel ne pourra pas être utilisé par la population handicapée mentale dans sa totalité. Les psychologues ont établi un ensemble de capacités et d'aptitudes que doit posséder l'utilisateur du logiciel:

- sensorielles : reconnaissance des formes et des objets;
- motrices : possibilité d'utiliser un clavier et une souris;

- intellectuelles : compréhension du langage parlé;
- culturelles : conscience qu'il faut dépenser de l'argent pour acheter quelque chose et capacité de reconnaître assez bien les pièces et billets;
- affectives : absence de blocages affectifs par rapport à l'argent.

2.3. LES DIFFERENTES FONCTIONS DU LOGICIEL DE GESTION DE BUDGET

Marc Déplechin et Gianni Strappazzon effectuent un stage en Suisse à La Castalie au cours duquel ils élaborent les spécifications du logiciel de gestion de budget avec Monsieur Baechler. Au cours de ces spécifications, cinq grandes fonctions sont définies :
- la tenue des comptes, qui est une gestion de l'argent a posteriori;
- le catalogue, qui permet à l'utilisateur de savoir ce qu'il peut acheter avec l'argent dont il dispose;
- le magasin, qui permet à l'utilisateur de simuler l'achat de différentes marchandises dans les limites de l'argent dont il dispose;
- l'épargne, qui est en fait la gestion de la tirelire de l'utilisateur;
- le budget, qui est une gestion de l'argent a priori.

2.4. LE LOGICIEL COMPTES : LA FONCTION "TENUE DES COMPTES"

Le logiciel Comptes n'est en fait qu'une partie du logiciel de gestion de budget. Il remplit la fonction de tenue des comptes. Il s'agit donc d'une comptabilité : l'utilisateur enregistre ses recettes et ses dépenses et obtient le solde de ces opérations qu'il compare avec la somme contenue dans son porte-monnaie. Cette fonction a été choisie par Marc Déplechin et Gianni Strappazzon comme premier sous-système utile car il semble indispensable de disposer d'une bonne tenue des comptes avant de pouvoir passer à des activités plus complexes comme la gestion de budget ou l'épargne. De

plus, pour réaliser un travail de qualité, ils ont préféré se limiter à ce seul sous-système.

2.5. LE PROGRAMME DE PARAMETRISATION ASSOCIE AU LOGICIEL COMPTES

La population des personnes handicapées mentales est fort hétérogène et le logiciel veut s'adresser à une large population. Pour réaliser cet objectif, le logiciel est conçu de manière à proposer parfois des éléments de dialogue différents pour la même fonctionnalité. Un programme de paramétrisation associé au logiciel Comptes permet, pour chaque individu, de sélectionner les éléments du dialogue adéquats et de n'utiliser que les fonctionnalités qui conviennent à la personne. Ce programme doit donc être employé par l'éducateur avant l'expérimentation pour paramétriser le logiciel selon l'individu.

2.6. LA LOGIQUE DU LOGICIEL COMPTES

Lors d'une session d'utilisation du logiciel, la personne handicapée mentale communique à l'ordinateur les recettes et les dépenses effectuées depuis la dernière session. Le logiciel peut ainsi calculer le solde de la session actuelle, qui sera considéré comme une recette lors de la session suivante. Cependant, il est tout à fait possible et même probable que l'utilisateur oublie de communiquer une recette ou une dépense. Dans ce cas, le solde ne correspondrait pas à la réalité et cette erreur se répercuterait de session en session. Pour remédier à cette situation, on demande à la personne de communiquer également la somme d'argent contenue dans son porte-monnaie. La comparaison de ce montant au solde de la session permet de faire apparaître un éventuel déséquilibre des comptes. Le cas échéant, l'utilisateur peut en chercher la cause et y remédier.

En fin de séance, le programme enregistre l'état du porte-monnaie qui est considéré comme le solde de la session et qui constituera donc une recette lors de la session suivante. Il a été considéré, en effet, que l'utilisateur a beaucoup moins de risques de se tromper lorsqu'il communique l'état de son porte-monnaie que lorsqu'il entre ses différentes recettes et dépenses. La logique du programme pour la séance t repose sur l'équation suivante :

solde + somme des recettes - somme des dépenses = état du porte-monnaie,

où - le solde est l'état du porte-monnaie enregistré lors de la séance t-1;
- l'état du porte-monnaie est la somme d'argent contenue dans le porte-monnaie au début de la séance t;
- la somme des recettes enregistrée à la séance t correspond à l'ensemble des recettes qui ont été perçues entre la séance t-1 et la séance t;
- la somme des dépenses enregistrée à la séance t correspond à l'ensemble des dépenses qui ont été effectuées entre la séance t-1 et la séance t.

2.7. LES FONCTIONNALITES DU LOGICIEL COMPTES

La première fonctionnalité est la saisie du nom de l'utilisateur. Ce nom permettra au programme d'accéder à la liste des paramètres qui correspond à cette personne. Ensuite, l'utilisateur enregistre la somme contenue dans son porte-monnaie. Après avoir saisi cette somme, le programme présente un écran d'état des comptes qui indique s'il y a équilibre ou pas à ce moment. Puis, il offre le choix entre les quatre options :

- l'enregistrement d'une recette subdivisé en quatre fonctionnalités :
  - la saisie du poste indiquant l'origine de la recette;

- la saisie du moment où la recette a été effectuée;
- la saisie du montant de la recette;
- la présentation de l'écran d'état des comptes;

- l'enregistrement d'une dépense également subdivisé en quatre fonctionnalités :
  - la saisie du poste pour lequel cette dépense a été faite;
  - la saisie du moment de cette dépense;
  - la saisie du montant de la dépense;
  - la présentation de l'écran d'état des comptes;

- la consultation de l'état du porte-monnaie qui contient :
  - la présentation de l'état du porte-monnaie;
  - la présentation d'un tableau récapitulatif des recettes et dépenses réalisées jusqu'à présent;
  - la présentation de l'écran d'état des comptes;

- l'option "fin" qui permet de quitter le programme après la présentation d'un écran d'état des comptes.

Après le déroulement de chaque option, sauf la fin, le programme revient au choix des quatre options.

2.8. L'INTERFACE DU LOGICIEL COMPTES

Pour chaque fonctionnalité du logiciel, nous décrivons l'interface utilisée.

1) LA SAISIE DU NOM

La question "Quel est votre nom ?" se présente à l'écran et l'utilisateur répond à l'aide du clavier.

Une demande de confirmation apparaît alors sous la forme d'une boîte de dialogue contenant l'interpellation "Est-ce bien juste ?" et deux boutons proposant les réponses "oui" ou "non". La personne "clique" alors sur le bouton de son choix.

2) LA SAISIE DE L'ETAT DU PORTE-MONNAIE

La question "Combien avez-vous d'argent ?" apparaît à l'écran et est prononcée avec un accent anglais par le synthétiseur vocal. En fonction de la valeur d'un paramètre, la saisie de la somme contenue dans le porte-monnaie se fait par souris ou par clavier.

- SAISIE PAR SOURIS :

La partie gauche de l'écran contient différents dessins de pièces et de billets qui sont des objets interactifs sur lesquels l'utilisateur peut "cliquer". Ces pièces et billets sont représentés de façon symbolique : un billet est un rectangle blanc à bords bleus à l'intérieur duquel est mentionnée la valeur du billet et une pièce est un cercle blanc à bord bleu à l'intérieur duquel la valeur est également indiquée. En dessous de ces billets et pièces se trouvent quatre autres boutons : "Efface", "Recommencer", "Fini" et une icône représentant une oreille.

Au centre de l'écran se trouve un thermomètre surmonté d'un montant qui est la somme d'argent maximale dont peut disposer la personne handicapée mentale et qui sert d'échelle à ce thermomètre. Dans la partie inférieure droite de l'écran, un rectangle affichant: "Total : ... Francs" comptabilise la somme déjà entrée.

Lorsque la personne handicapée mentale "clique" sur une pièce ou un billet, une trace de celui-ci apparaît dans la partie droite de l'écran (mais

avec une dimension plus petite), le total augmente en conséquence, le thermomètre se remplit en bleu proportionnellement à la somme enregistrée et un repère jaune vient se mettre à son sommet, enfin le synthétiseur vocal prononce le montant du billet ou de la pièce .

Lorsque la personne handicapée mentale "clique" sur l'icône de l'oreille, la dernière phrase qui a été prononcée est répétée.

Si elle sélectionne un des trois boutons "Efface", "Recommencer" ou "Fini", une boîte de dialogue apparaît pour demander une confirmation de son intention à l'utilisateur .

Le bouton "Efface" permet d'effacer le dernier billet ou la dernière pièce entrée. Le bouton "Recommencer" permet d'effacer tout ce qui a déjà été enregistré et l'option "Fini" de terminer la saisie.

Cet écran est présenté sur la planche "Saisie de l'état du porte-monnaie par billets (Première version du logiciel)" dans la section 4.3.

- SAISIE PAR CLAVIER :

L'utilisateur tape les chiffres correspondant au montant qu'il veut communiquer.

Lorsque la personne handicapée mentale tape sur la touche "return", le niveau du thermomètre monte et une boîte de dialogue apparaît et propose deux boutons : "Efface" et "Fini". La personne "clique" sur un des deux boutons à l'aide de la souris.

3) LA PRESENTATION DES DIFFERENTES OPTIONS

La phrase "Choisissez une activité" apparaît en haut de l'écran et est prononcée par la synthèse vocale avec un accent anglais. Sous cette phrase apparaissent quatre icônes : "Recette", "Dépense", "Etat du porte-monnaie"

et, en dessous, "Fin". La personne effectue son choix dans le menu grâce à la souris.

4) LA SAISIE DU POSTE D'UNE RECETTE

La question posée oralement et par écrit est la suivante : "D'où vient cette recette ?". La personne a le choix entre quatre réponses : parents, argent de poche, travail et tirelire représentés par des icônes. De nouveau, une confirmation est exigée .

5) LA SAISIE DU MOMENT D'UNE RECETTE (CYCLE D'UN JOUR)

La question "Quand avez-vous fait cette recette ?" est posée à l'individu par écrit et oralement. Trois icônes représentant le matin, l'après-midi et le soir sont présentées à la personne qui choisit à l'aide de la souris. Une fois son choix posé, une boîte de dialogue de confirmation apparaît.

6) LA SAISIE DU JOUR D'UNE RECETTE (CYCLE D'UNE SEMAINE)

Cette saisie se présente comme la précédente, mais les trois icônes représentant les moments de la journée sont remplacées par sept icônes correspondant aux jours de la semaine. Le jour actuel figure en blanc, les jours antérieurs figurent en vert et les jours postérieurs en rouge.

7) LA SAISIE DU MONTANT D'UNE RECETTE

Pour cette saisie, il est possible de se référer à la saisie du porte-monnaie. Nous présentons les différences ci-après.

- La question de départ est celle-ci : "Combien avez-vous reçu d'argent ?".
- Une icône "Recette" se trouve sur l'écran.
- Le thermomètre monte en vert et la barre jaune reste fixe car il s'agit d'un repère indiquant le niveau du porte-monnaie. La partie verte devient ensuite bleue.
- Pour la saisie par souris, le bouton "Quitter" remplace le bouton "Recommencer" et permet de ne pas enregistrer la recette.

- Pour la saisie par clavier, ce bouton "Quitter" est ajouté aux deux autres ("Efface" et "Fini").

Tout ce qui concerne les dépenses se passe pratiquement comme pour les recettes. Par conséquent, nous ne signalons que les différences.

8) La saisie du poste d'une depense

L'interrogation de départ devient: "D'où vient cette dépense ?" et les postes représentés par des icônes sont : nourriture, tabac, vêtements, transports, magazines, tirelire, loisirs et choses diverses.

9) La saisie du moment d'une depense (cycle d'un jour ou d'une semaine)

La question est cette fois : "Quand avez-vous fait cette dépense?".

10) La saisie du montant d'une depense

- La première interpellation devient :"Combien avez-vous dépensé ?".
- L'icône présente dans cet écran est bien sûr celle des "Dépenses".
- Le thermomètre fonctionne différemment : il se colore en rouge en descendant à partir de la limite supérieure du niveau bleu. Une fois que l'utilisateur a terminé la saisie, cette partie rouge s'efface.

11) La presentation de l'etat du porte-monnaie

Cette présentation peut être alphanumérique ou graphique.

Si la présentation est alphanumérique, la phrase "Il vous reste ..... francs" est affichée et prononcée. Un thermomètre indique le niveau des comptes. Lorsqu'on "clique" sur le bouton "Fini", si les comptes ne sont pas équilibrés, une boîte de dialogue avertit la personne de l'importance et du sens du déséquilibre.

Si la présentation est graphique, tout se passe de la même façon si ce n'est que la somme enregistrée est présentée sous forme de billets et de pièces symboliques tels qu'ils ont été décrits dans l'écran de saisie par billets.

12) La presentation du tableau recapitulatif des recettes et depenses

Un premier écran présente les recettes. Il contient un thermomètre et quatre colonnes : Jour/Moment, Poste, Montant, Total. Les recettes s'affichent une à une et le niveau du thermomètre monte en même temps. Lorsque toutes les recettes sont affichées, un bouton "Suite" apparaît.

Le deuxième écran présente les dépenses et tout se déroule de la même manière.

13) La presentation d'un ecran d'etat des comptes

Cet écran présente le thermomètre, qui indique l'état des comptes, et une balance. Celle-ci n'est en équilibre que si les comptes sont justes. Cette balance porte sur le plateau gauche l'initiale R pour Recettes, et sur le droit D pour Dépenses. Elle penche vers la gauche (du côté des recettes) si l'état du porte-monnaie est inférieur à la somme des recettes moins la somme des dépenses, et vers la droite dans le cas contraire. De plus, une phrase signale qu'"Il manque ... Francs de dépenses" dans le premier cas et qu'"Il manque ... Francs de recettes" dans le second. Lorsque la balance est en équilibre, la phrase "Bravo, il y a équilibre des comptes" apparaît. Un bouton "Fin" permet de passer à l'écran suivant.

14) Sortie du programme

Lorsqu'on choisit l'option "Fin", on se trouve face à l'écran de présentation de l'état des comptes, celui-ci propose en outre la question "Voulez-vous quitter le programme ?".

CHAPITRE III : EVALUATION DE LA PREMIERE VERSION DU LOGICIEL DE TENUE DE COMPTES

" Le créateur nous a donné deux oreilles et une bouche, pour que nous écoutions deux fois plus que nous ne parlions"

Zénon d'Elée

3.1. LES OBJECTIFS DE L'EVALUATION

Pour notre évaluation, il nous a semblé primordial de commencer par s'interroger sur l'utilité du logiciel par rapport à ses objectifs. Il convient de vérifier d'abord si le projet conçu est opératoire, s'il correspond aux objectifs fixés. En effet, il serait inutile de s'intéresser à l'interface du logiciel ou à la durée d'apprentissage, par exemple, si on arrivait à la conclusion qu'il ne sert à rien.

En nous basant sur cette idée, nous avons dressé une liste d'éléments à évaluer subdivisée en 4 groupes :

- les éléments liés à l'utilité du programme de tenue de comptes en tant qu'outil d'autonomie;
- les éléments liés à l'utilité du programme en tant qu'outil d'apprentissage;
- les éléments liés aux fonctionnalités et à l'interface du logiciel, pour lesquels on prend plutôt une vision de concepteur qui analyse de façon critique son travail ainsi que les choix qu'il a pris;
- les éléments psychologiques divers pouvant apporter des informations utiles aux psychologues.

Nous précisons que cette liste n'est pas exhaustive et que d'autres objets d'évaluation pourraient y être ajoutés.

L'UTILITE DU PROGRAMME DE TENUE DE COMPTES EN TANT QU'OUTIL D'AUTONOMIE

1) Le logiciel permet-il à une personne handicapée mentale de tenir ses comptes de manière autonome ?

Il s'agit donc de vérifier si, d'une part, le logiciel est effectivement un **outil de tenue des comptes** et si, d'autre part, une personne handicapée mentale peut vraiment l'utiliser pour tenir ses comptes de manière **autonome.**

Ce dernier élément contient deux aspects :

- un aspect plutôt technique lié à la capacité de manipuler la souris, le clavier, à la capacité de passer d'un écran au suivant, de communiquer son nom,...
- un aspect plutôt lié à la réalité, à la capacité de mener à bien l'activité de tenue des comptes. Il s'agit de vérifier si la personne choisit bien l'option "recette" pour communiquer le fait qu'elle a reçu de l'argent, si elle enregistre bien la somme contenue dans son porte-monnaie, si elle est capable de se rappeler des dépenses qu'elle a faites durant la semaine,...

2) Quelle est l'utilité du logiciel Comptes ?

Il ne suffit pas de vérifier si le logiciel permet à une personne handicapée mentale de tenir ses comptes de manière autonome, il faut encore voir si ce moyen présente des avantages "objectifs" par rapport aux autres moyens de gestion et si ces avantages sont plus importants que les inconvénients. On se pose donc la question de savoir si le logiciel apporte **objectivement** quelque chose à la personne handicapée mentale.

3) La personne handicapée mentale éprouve-t-elle un certain plaisir à tenir ses comptes avec le logiciel?

S'il apparaît que l'utilisateur éprouve un réel plaisir à se servir de l'ordinateur pour tenir ses comptes, cela constitue un argument supplémentaire en faveur du logiciel par rapport à l'utilisation d'autres méthodes.

D'autre part, s'il apparaît que l'individu ne ressent aucun plaisir ou même qu'il éprouve une certaine angoisse face à l'ordinateur, il est inutile d'espérer qu'il emploie le logiciel.

4) Le logiciel peut-il inciter des personnes handicapées mentales à tenir leurs comptes ?

On peut se demander si le logiciel ne pourrait pas encourager certaines personnes handicapées mentales à tenir leurs comptes avec le logiciel Comptes alors qu'elles n'auraient jamais **voulu** le faire autrement. Il ne s'agit plus ici d'évaluer une capacité mais une volonté.

L'UTILITE DU PROGRAMME EN TANT QU'OUTIL D'APPRENTISSAGE

5) Le logiciel permet-il d'apprendre de nouveaux termes ?

Le logiciel utilise un certain nombre de termes tels que "recette", "dépense", "état du porte-monnaie", "solde",...
Peut-être permet-il à certaines personnes handicapées mentales d'étendre leur vocabulaire à ces mots ?

6) Le logiciel permet-il de comprendre de nouveaux concepts ?

Il faut bien faire la distinction entre cet élément et le précédent. En effet, apprendre un nouveau terme correspond à lier un nouveau mot à un concept déjà connu. Par contre, ici, il s'agit vraiment d'acquérir une nouvelle notion même si on ne sait pas nécessairement y faire correspondre le terme adéquat.

7) Le logiciel permet-il des apprentissages au niveau des acquis de la personne handicapée mentale ?

Les acquis en lecture, écriture, calcul,... peuvent-ils évoluer à plus ou moins long terme ?

8) Le logiciel peut-il entraîner des changements dans les comportements des individus vis-à-vis de l'argent ?

Les personnes handicapées mentales peuvent-elles changer d'attitude vis-à-vis de l'argent ? Prennent-elles conscience de son importance dans la vie quotidienne? Réagissent-elles ensuite à cette éventuelle prise de conscience par une volonté de le gérer ?

9) Le logiciel permet-il aux personnes handicapées mentales d'acquérir une meilleure notion des valeurs ?

Nous désirons savoir si de nouveaux acquis concernant la notion du prix des choses, de l'état du porte-monnaie, de la valeur relative des pièces et des billets,... peuvent apparaître.

L'INTERFACE ET LES FONCTIONNALITES DU LOGICIEL

10) Les fonctionnalités offertes par le logiciel sont-elles suffisantes et satisfaisantes ?

Si le logiciel permet à une personne handicapée mentale de tenir ses comptes, on peut se demander si certaines modifications des fonctionnalités existantes ou la création de nouvelles fonctionnalités ne lui faciliteraient pas la tâche.

11) L'interface réalisée dans le logiciel convient-elle bien aux personnes handicapées mentales ou devrait-elle être améliorée ?

Cette partie se propose d'examiner l'ergonomie générale du logiciel et de critiquer les éléments de l'interface.

LES INFORMATIONS PSYCHOLOGIQUES DIVERSES

12) Quelle est la durée d'apprentissage du logiciel ?

Il serait intéressant de savoir si des progrès significatifs se font assez rapidement et de déterminer le moment où on peut dire que le logiciel est "maîtrisé" par la personne handicapée mentale.

13) Quel type de personnes handicapées mentales peut utiliser le logiciel Comptes ?

La détermination des prérequis nécessaires à l'utilisation d'un logiciel est une opération difficile. Colette Legrand, dans sa thèse "Informatique et handicap" défendue à l'Université de Technologie de Compiègne, propose la démarche suivante.

Le psychologue doit situer le futur logiciel dans un premier référentiel avant même sa conception.

Ce premier référentiel est exprimé par la question: "Quel est le contenu du logiciel au niveau perceptif, moteur, cognitif et affectif ?".
Pour le logiciel Comptes, on pourrait répondre :
- niveau perceptif : important. Le programme exige en effet que la personne handicapée puisse reconnaître les formes et les couleurs;
- niveau moteur : relativement important. Il faut pouvoir utiliser un clavier d'ordinateur et savoir manipuler une souris;
- niveau cognitif : très important;
- niveau affectif : un peu (relation à l'argent).

C'est surtout au niveau cognitif que les prérequis sont les plus importants. Le psychologue pourra dès lors préciser ces prérequis en analysant le logiciel avec le référentiel "Opérations cognitives": "Que contient le logiciel en reconnaissance de formes, maîtrise de l'espace, maîtrise du temps, expression verbale et écrite, calcul, raisonnement ?".

Même en suivant cette démarche, il est probable que le psychologue ne pourra pas déterminer précisément les prérequis. Il convient donc de les évaluer.

Après l'expérimentation du programme, on peut :
- vérifier qu'on n'a pas sur-estimé ou sous-estimé l'importance d'un prérequis;

- essayer de diminuer l'importance des prérequis les plus contraignants de façon à pouvoir développer finalement le produit le plus ouvert possible à la population handicapée.

14) Les effets imprévus

Le logiciel pourrait avoir certains effets tout à fait inattendus. Il serait intéressant de les mentionner.

Remarque : Evaluation des utilisateurs

Il est également possible d'évaluer les individus mais ceci n'est pas le but de notre évaluation. Cela peut être très intéressant pour des éducateurs ou des psychologues.

Un individu apprend-il rapidement? Se base-t-il plutôt sur tel ou tel élément de l'interface? Rencontre-t-il un problème particulier? Maîtrise-t-il particulièrement bien un élément?

3.2. LES CONDITIONS D'EXPERIMENTATION

Afin d'évaluer le logiciel Comptes, nous avons défini le cadre dans lequel l'expérimentation prendrait place.

Pour disposer d'un échantillon assez représentatif de la population, nous désirons expérimenter le logiciel avec environ 50 personnes handicapées mentales (approximativement 15 en Belgique, 15 en Suisse et 15 en France).

Les 50 utilisateurs ne peuvent pas, pour des raisons matérielles, provenir d'une seule et même institution et il est d'ailleurs souhaitable qu'il n'en soit pas ainsi de manière à pouvoir évaluer l'outil de tenue des comptes dans des environnements comparables mais différents.

Nous prenons donc contact avec une série d'institutions intéressées dans lesquelles un éducateur est responsable de l'expérimentation avec "ses" utilisateurs. Ces utilisateurs sont choisis en fonction des prérequis fixés pour le logiciel, de leurs motivations et de leurs disponibilités. Ils devront utiliser au minimum 5 fois le logiciel à raison d'une séance par semaine.

Nous familiarisons les différents éducateurs avec l'ordinateur Amiga et avec le logiciel Comptes. Nous leur expliquons ensuite les objectifs poursuivis par le programme et par l'évaluation ainsi que la marche à suivre pour l'expérimentation.

Nous avons décidé de leur demander, dans la mesure du possible, de paramétriser le programme Comptes de manière à ce qu'il présente un maximum d'objets interactifs et de fonctionnalités. De cette manière, nous pouvons évaluer un grand nombre d'interfaces et de fonctionnalités. De plus, cela contribue à l'homogénéité de notre expérimentation.

3.3. LES OUTILS D'EVALUATION

Nous présentons dans cette section les outils d'évaluation utilisés pour répondre aux objectifs de l'évaluation.

Par ailleurs, nous avons montré la nécessité de réaliser une expérimentation afin d'évaluer un logiciel pour personnes handicapées mentales (cfr. 1.3.). Le premier outil que nous présentons, l'analyse du logiciel sur base de la théorie de l'interface homme-machine, ne s'inscrit pas obligatoirement dans le cadre d'une expérimentation. Par contre, les quatre autres outils à savoir les questionnaires, les traces d'utilisation, les entrevues avec les éducateurs et les contacts avec les utilisateurs exigent une expérimentation. Afin de classer ces derniers, nous avons défini une grille de positionnement des outils.

	avec l'intervention de l'accompagnateur	sans l'intervention de l'accompagnateur
"formel"		
"non formel"		

L'axe des abscisses répond à la question : "L'accompagnateur (éducateur, psychologue, logopède,...) intervient-il de façon directe dans l'utilisation de l'outil ?".

L'axe des ordonnées répond à la question : "Existe-t-il une méthode formelle permettant de tirer des résultats des expérimentations faites avec l'outil ?".

L'étude de ces deux axes permet de mettre en évidence, avant même d'obtenir les résultats de l'expérimentation, certains avantages et inconvénients propres à chaque outil. Par exemple, on peut raisonnablement supposer que les outils qui nécessitent l'intervention de l'accompagnateur introduiront un biais dû à l'appréciation de cet accompagnateur. Rappelons toutefois qu'il est impossible de concevoir une évaluation de logiciels pour personnes handicapées mentales sans son aide. De même, les données récoltées grâce aux outils "non formels" seront difficilement exploitables de manière rigoureuse.

Chaque outil a ses propres limites et il est donc souhaitable d'en utiliser plusieurs pour répondre à chaque objectif de l'évaluation de façon à pallier à leurs lacunes respectives.

Par ailleurs, le rôle de deux acteurs de l'évaluation est précisé grâce à cette grille. Ainsi, le premier axe montre le rôle de l'éducateur dans la phase d'expérimentation et le second, le rôle du statisticien dans la phase d'exploitation des résultats.

Nous pouvons situer les quatre outils qui nécessitent une expérimentation dans la grille de positionnement:

	avec l'intervention de l'accompagnateur	sans intervention de l'accompagnateur
"formel"	questionnaires	traces d'utilisation
"non formel"	entrevues avec l'éducateur	contacts directs avec les personnes handicapées

Nous verrons par la suite que tous les autres outils d'évaluation imaginés peuvent également se situer par rapport à notre grille de positionnement.

3.3.1. L'INTERFACE HOMME-MACHINE.

3.3.1.1. Définition

- L'interface homme-machine règle le dialogue entre l'utilisateur et l'ordinateur (Petaud, 1986).

- Une interface est un système informatique utilisé par une personne pour réaliser une tâche accomplie à l'aide d'un ensemble de moyens informatiques (cours d' "Interface homme-machine" donné par Monsieur F. Bodart, 1989).

Cette définition nous paraît plus opérationnelle. En effet, une bonne interface est celle qui permet à l'utilisateur de réaliser le mieux possible la tâche qu'il doit accomplir.

3.3.1.2. Utilisation de l'outil pour répondre aux objectifs de l'évaluation

Evidemment, l'utilisation des concepts et recommandations en matière d'interface homme-machine ne nous permet pas directement d'évaluer si le programme Comptes répond bien aux objectifs qu'il poursuit (à savoir, notamment fournir un outil de tenue de comptes utilisable par une personne handicapée mentale de façon autonome).

Mais, s'il apparaissait que les objectifs du programme ne sont pas ou sont mal vérifiés et que les fonctionnalités du programme sont pourtant correctement définies, l'interface homme-machine pourrait nous aider à redéfinir les interfaces abstraites (les composants du dialogue) et ensuite les interfaces concrètes (la présentation).

Si, par contre, il apparaissait que les objectifs poursuivis par le logiciel sont atteints dans une large mesure, l'interface homme-machine nous permettrait de combler certains écarts entre ces objectifs et ce que le logiciel donne réellement la possibilité de faire et/ou d'utiliser le logiciel avec une plus large partie de la population handicapée .

3.3.2. LES QUESTIONNAIRES

3.3.2.1. Définition

"*L'enquête par questionnaire est un instrument de prise de l'information, basé sur l'observation et l'analyse de réponses à une série de questions posées.*"
Jean-Pierre Pourtois et Huguette Desmet définissent de cette manière l'enquête par questionnaire dans "Epistémologie et instrumentation en sciences humaines".

3.3.2.2. Le déroulement d'une enquête par questionnaire : application à l'évaluation du logiciel Comptes

Les auteurs précités proposent également une démarche à suivre pour concevoir une enquête par questionnaire de façon à en extraire des informations crédibles. Nous reprenons ici les 9 phases qu'ils exposent et nous montrons comment elles se sont déroulées concrètement au cours de l'évaluation du logiciel Comptes.

1) La détermination des objectifs et des hypothèses de recherche

Dans leur mémoire, Marc Déplechin et Gianni Strappazzon préconisent deux grands types d'évaluations : une évaluation de l'interface qui porte principalement sur le dialogue et les fonctionnalités choisis et une évaluation pédagogique qui concerne plutôt l'utilité du programme, les bénéfices que les personnes handicapées retirent de son utilisation,...

Nous avons préféré préciser davantage nos objectifs d'évaluation. Ceux-ci ont d'ailleurs été exposés à la section 3.1.

Remarquons cependant que l'enquête par questionnaire n'est pas notre seul moyen d'évaluation et que certains objectifs ne peuvent pas être traités par questionnaire. Pour ceux-ci, d'autres outils d'appréciation sont nécessaires. Tous ces objectifs ne sont pas définis uniquement pour notre enquête par questionnaire mais bien pour l'ensemble de l'évaluation.

2) La construction du questionnaire

Un premier questionnaire a été élaboré par des psychologues, des éducateurs et des informaticiens. Les psychologues et les éducateurs ont pris en charge la partie liée à l'évaluation pédagogique et les informaticiens se sont penchés sur celle qui concerne l'évaluation de l'interface. Il est peut-être préférable que ces spécialistes se limitent aux domaines pour lesquels ils sont les plus compétents. Ainsi, ni l'informaticien, ni le psychologue, ni l'éducateur ne peuvent prétendre posséder les connaissances et les modèles théoriques nécessaires à la construction de la totalité du questionnaire. Une collaboration est donc absolument indispensable: une approche pluridisciplinaire est ici encore nécessaire.

3) L'établissement de l'échantillon

Nous désirons obtenir un échantillon de 50 individus présentant un handicap mental léger ou modéré. Ces individus doivent satisfaire aux prérequis établis (cfr. 2.1). Ils sont issus d'institutions belges, suisses et françaises avec lesquelles nous prenons contact.

4) Le test du questionnaire auprès d'un sous-échantillon

Avant l'expérimentation proprement dite, il convient de s'assurer auprès d'un sous-échantillon représentatif des éducateurs que le questionnaire est compris, qu'il est complet et qu'il ne comporte pas de questions inutiles.

Marc Déplechin et Gianni Strappazzon ont entamé une expérimentation du logiciel Comptes sur un petit échantillon. Nous avons recueilli les résultats de celle-ci et nous avons ainsi pu modifier le questionnaire. Cette expérimentation nous a donc servi de test du questionnaire et nous a permis d'en élaborer une nouvelle version sur base de laquelle l'expérimentation proprement dite aura lieu.

Afin de récolter des critiques sur le questionnaire, nous aurions également pu distribuer aux éducateurs un "questionnaire sur le questionnaire" tel que celui que nous avons rédigé et que le lecteur pourra trouver en annexe 3.

5) La réalisation de l'enquête auprès de l'échantillon retenu

Nous nous sommes rendus dans les différentes institutions avec lesquelles nous avions décidé de mener l'expérimentation. Nous y avons rencontré les différents éducateurs et nous leur avons présenté le logiciel et son fonctionnement, le questionnaire et ce que nous attendions de l'expérimentation.

Nous les avons ensuite laissés mener l'expérimentation et nous sommes retournés pour rassembler les questionnaires remplis. Mais cette phase ne s'est pas déroulée aussi simplement qu'il n'y paraît comme nous le verrons dans la section 3.4 dans laquelle nous exposons les problèmes que nous avons rencontrés.

6) Le codage du matériel récolté

Afin de pouvoir présenter de manière synthétique les réponses aux questionnaires et pour pouvoir les traiter facilement, nous avons décidé de les représenter de manière numérique.

Chaque question est numérotée de manière à être rapidement identifiable.

Puisque les questions offrent au maximum 3 choix, les réponses possibles sont représentées par 1, 1/2 ou 0.

La convention que nous avons prise est la suivante :

- pour des questions à connotation positive (Ex.: "La personne sait lire ?"), une réponse positive ("Oui", "Parfaitement",...) est représentée par le chiffre 1 et une réponse négative ("Non", "Pas du tout",...) par le chiffre 0;
- pour des questions ayant une connotation négative (Ex.: "La personne ne sait pas lire ?"), c'est le contraire;
- dans les deux cas, une réponse "intermédiaire" ("A moitié", "plus ou moins",...) est représentée par 0.5.

Notons qu'une question à connotation négative est une question pour laquelle une réponse positive signifie que l'utilisateur n'a pas compris un élément, a commis une erreur,...

La numérotation des questions ainsi que le codage utilisé pour chacune d'entre elles sont repris dans le questionnaire figurant à l'annexe 1.

7) Le traitement des données

Il existe une grande variété de méthodes qui permettent de traiter les données récoltées. Un questionnaire se prête d'ailleurs très bien aux traitements quantitatifs. Nous en avons appliqué quelques-unes que nous présentons au point 3.3.2.4. Le statisticien nous apportera une aide utile au cours de cette étape.

8) L'interprétation et la présentation des résultats

Nous avons interprété plusieurs résultats avec l'aide de psychologues et d'éducateurs. Nous présentons nos conclusions à la section 3.5.

9) La vérification de la fidélité et de la validité des données

Nous avons confronté les données que nous avons obtenues grâce aux questionnaires à celles que nous avons pu acquérir à l'aide d'autres outils d'évaluation. Ceci nous a permis de vérifier la cohérence et la concordance de nos données. Il est très souvent préférable de disposer de plusieurs sources d'information car chacune présente des avantages et des inconvénients. De plus, la confrontation de ces sources d'information permet de valider les données ainsi recueillies. Nous évaluerons d'ailleurs les outils d'évaluation eux-mêmes dans la section 3.6.

3.3.2.3. Présentation du questionnaire d'évaluation du logiciel Comptes

L'enquête par questionnaire que nous avons menée est très particulière. En effet, le questionnaire n'est pas complété par la personne directement concernée, en l'occurence la personne handicapée mentale, mais par un intermédiaire qui est le plus souvent un éducateur.

Le questionnaire est en fait constitué d'un ensemble de questions pour lesquelles nous attendons une réponse. L'éducateur peut choisir les moyens qu'il mettra en oeuvre pour obtenir une réponse qui reflète la réalité. Nous lui laissons la liberté d'obtenir la réponse comme il le veut. Cela ressort de sa responsabilité. Les réponses à certaines questions peuvent s'obtenir en observant simplement ce qu'il se passe durant une session d'utilisation (Ex.: "La personne clique sur "fin" par inadvertance"). Toutefois, d'autres questions demandent une appréciation de la part de l'éducateur (Ex. : "Les termes utilisés sont compris par la personne"). Dans ce cas, il est nécessaire que celui-ci se forge une opinion en posant des questions à l'utilisateur ou en lui faisant faire un exercice par exemple. Avec cet outil, l'éducateur joue donc un rôle d'évaluateur.

Comme nous l'avons déjà vu, nous avons construit ce questionnaire à partir d'une première version réalisée pour l'expérimentation lancée par Marc Déplechin et Gianni Strappazzon. La nouvelle version du questionnaire comporte 6 parties et est présentée en annexe 1.

La première partie du questionnaire (pages 1 et 2) est adressée à l'éducateur et contient 4 questions ouvertes qui concernent les objectifs que se sont fixés les éducateurs eux-mêmes quant à l'utilisation du programme avec des personnes handicapées mentales. Cette partie doit être remplie

une seule fois, avant de commencer l'expérimentation, et n'a pas été modifiée par les psychologues par rapport à la première version du questionnaire.

La deuxième partie du questionnaire (pages 3 à 7) est appelée la "fiche signalétique de l'utilisateur". Elle comprend, d'une part, quelques questions qui permettent d'identifier la personne (son nom, le nom de l'institution dont elle fait partie,...) et, d'autre part, une série de questions fermées visant à connaître les capacités de l'utilisateur et qui constitue le "questionnaire psychologique". Cette grille a subi quelques modifications suite à la première expérimentation du logiciel.

Premièrement, nous avons modifié les rubriques à l'intérieur desquelles étaient réparties ces questions : nous avons subdivisé certaines rubriques car elles ne nous semblaient pas suffisamment précises. Les questions sont maintenant classées dans les rubriques suivantes : acquis en lecture, en écriture, en calcul, notion de temps, notion de l'argent, notion du prix des choses, notion de l'état du porte-monnaie, attention portée à l'argent et familiarité avec l'ordinateur.

Ensuite, alors que toutes ces questions étaient posées avant et après l'expérimentation, nous avons décidé que certaines ne seraient plus posées après l'expérimentation. En effet, d'une part, certains éducateurs s'étaient plaints de la longueur du questionnaire et, d'autre part, en discutant avec une psychologue, il est apparu qu'il n'y avait que très peu de chances d'évolution des connaissances dans certains domaines sur la durée de l'expérimentation (c'est-à-dire environ un mois). Nous avons donc profité de cette occasion pour alléger quelque peu le questionnaire en ne posant plus qu'avant l'expérimentation les questions concernant les acquis en lecture, écriture et calcul et la notion de temps.

Nous aurions également souhaité réduire le questionnaire en ne posant plus qu'une seule question pour des rubriques telles que "acquis en lecture", "acquis en écriture", "acquis en calcul",... Nous aurions ainsi remplacé des questions comme " La personne lit son nom, lit des mots, lit des phrases courtes,..." par une seule question ("La personne sait lire") à laquelle l'éducateur aurait répondu en donnant par exemple une cote comprise entre 0 et 10. Nous avons finalement abandonné cette idée car le questionnaire devait être rempli par différents éducateurs qui auraient pu coter de façons très différentes. Si nous voulions éviter ce problème, nous

aurions dû définir précisément pour chaque question à quoi correspondait la cote. Le questionnaire n'aurait sûrement pas été allégé.

Ensuite, nous avons remplacé certaines questions à choix dichotomique (Oui ou Non) par des questions à choix multiples offrant 3 possibilités de réponse car nous avions remarqué que certaines questions étaient trop tranchées et que des éducateurs éprouvaient des difficultés à y répondre de façon binaire.

Enfin, nous avons supprimé et ajouté quelques questions. Nous avons, par exemple, scindé la question "Comprend le sens de "recette" et "dépense" " en deux questions car il nous semble que l'on peut comprendre le sens de l'un de ces deux termes sans nécessairement connaître l'autre. Nous avons également demandé si la personne conserve son ticket de caisse car il est apparu au cours de la première expérimentation que certains individus ont commencé à garder leurs tickets grâce au logiciel. Ceci était tout à fait inattendu et nous semble être un bon indice de l'intérêt que la personne porte à sa tenue des comptes.

La troisième partie du questionnaire (pages 8 à 12) est une grille appelée "questionnaire ergonomique" qui contient des questions fermées à choix dichotomiques ou multiples (à 3 possibilités) visant essentiellement à évaluer l'interface du logiciel. Cette grille doit être complétée par l'éducateur après chaque séance d'utilisation du logiciel de manière à pouvoir mesurer précisément le moment où d'éventuels progrès apparaîtront. Nous l'avons également modifiée en proposant des choix multiples (3) plutôt que dichotomiques pour certaines questions et en supprimant des questions ou en ajoutant des questions qui semblaient utiles. Suite à la première expérimentation, nous avons pu nous rendre compte des éléments les plus problématiques et poser des questions en conséquence pour cerner les difficultés rencontrées. Nous avons également pu remarquer que certaines questions avaient un sens très proche et des réponses fort corrélées : elles pouvaient donc être "fusionnées".

La quatrième partie du questionnaire (pages 13 à 16) doit être complétée lorsque l'expérimentation est terminée. Elle comprend des questions ouvertes permettant à l'éducateur de critiquer le logiciel, de proposer des améliorations, de dire si le logiciel a pu satisfaire à ses objectifs et de proposer des prérequis jugés nécessaires pour pouvoir utiliser le logiciel. Nous n'avons pas modifié cette partie.

La cinquième partie (pages 17 et 18) est destinée à l'utilisateur. L'éducateur doit poser quelques questions à l'utilisateur et consigner ses réponses. Il s'agit de questions ouvertes. Cette partie doit être remplie en fin d'expérimentation. L'objectif poursuivi est d'approcher ce que la personne handicapée mentale pense elle-même du logiciel; c'est elle qui évalue.

Enfin, nous avons ajouté une dernière partie à ce questionnaire qui est constitué d'un certain nombre de pages contenant chacune le dessin d'un écran ainsi qu'une plage blanche sur laquelle l'éducateur peut consigner toutes ses critiques et suggestions concernant l'écran. Tous les écrans du logiciel sont repris dans ce document. Ceci permet donc à l'éducateur d'indiquer sur le dessin même de l'écran les éléments de l'interface qui lui semblent être problématiques. De plus, en ayant chaque écran sous les yeux, l'éducateur risque moins d'oublier de mentionner certaines critiques. Cette partie ne doit pas être remplie pour chaque utilisateur.

3.3.2.4. Utilisation de l'outil pour répondre aux objectifs de l'évaluation

Le questionnaire peut être utilisé pour traiter pratiquement tous les objectifs d'évaluation que nous nous sommes fixés car il contient des questions relatives à la plupart de ces objectifs. Nous pouvons, par exemple, utiliser les deux grilles contenues dans le questionnaire pour répondre à plusieurs objectifs liés à l'utilisation du logiciel en tant qu'outil d'apprentissage ainsi que pour tirer certaines conclusions à propos de l'interface du logiciel.

Par ailleurs, certaines questions posées à l'utilisateur peuvent nous indiquer si la personne éprouve un certain plaisir à utiliser le logiciel.

Cependant, il nous semble que le questionnaire est un outil assez peu approprié pour répondre à plusieurs questions que nous nous posons à propos de l'utilité du logiciel en tant qu'outil d'autonomie pour tenir ses comptes. Dans ce cas, des outils d'appréciation moins formels semblent plus adéquats.

3.3.2.5. Les méthodes d'exploitation du questionnaire

Nous exposons quelques méthodes que nous avons utilisées pour traiter les données du questionnaire en précisant chaque fois comment nous les avons employées.

3.3.2.5.1. L'analyse par la statistique inférentielle

Il est possible de traiter les réponses aux questionnaires grâce à la statistique inférentielle. En fait, nous voudrions procéder à deux types d'analyses :

- nous aimerions estimer la proportion de réponses positives ou négatives à une certaine question pour la population. Pour un certain nombre de questions, cela pourrait nous apporter de l'information utile;

- nous aimerions également estimer la proportion de personnes qui peuvent progresser dans la population sur une expérimentation d'une certaine durée. Cela peut être d'un grand intérêt pour vérifier si le logiciel peut bien jouer le rôle d'un outil d'apprentissage.

Dans les deux cas, il faudrait donc chercher un estimateur des proportions (les valeurs estimées), qui nous donnerait une valeur vraisemblable pour la population. Cependant, cette information paraît insuffisante et nous aimerions connaître des limites pour notre estimation, c'est-à-dire que nous souhaiterions construire un intervalle de confiance pour notre estimateur, avec un certain niveau d'incertitude α fixé. Cela signifie donc que la probabilité que l'intervalle de confiance construit comprenne la valeur estimée est de $1-\alpha$.

Si l'effectif de l'échantillon était suffisamment grand, nous pourrions nous baser sur une distribution normale mais ce n'est pas le cas et nous avons donc dû nous baser sur une autre distribution : la binomiale.

Nous la présentons brièvement sur base des cours de "Probabilité" et d' "Introduction à la statistique mathématique" de Madame Noirhomme. Nous verrons ensuite pourquoi cette distribution convient bien à notre analyse.

Soit w, l'échantillon;
n, l'effectif de l'échantillon, c'est-à-dire le nombre d'individus pris en considération;
S(w), le nombre de succès dans l'échantillon w;
p, la probabilité d'un succès pour la population considérée.

Alors, le nombre de succès S, pour n observations, a une distribution binomiale de paramètres (n, p) :

$$P[\, S(w) = k] = C_n^K * p^K * (1-p)^{n-k}$$

Si nous posons que le niveau d'incertitude α est de 0.05, nous pouvons consulter une table basée sur la distribution binomiale pour obtenir p1, la limite inférieure de l'intervalle de confiance, et p2, la limite supérieure. Cette table se trouve en annexe 5. Les entrées dans la table sont le niveau d'incertitude α, le nombre de succès dans l'échantillon S(w) et l'effectif de l'échantillon n. On obtient alors l'intervalle $p1 \leq p \leq p2$, et

$$Pr\,[\, p1 \leq p \leq p2] \;=\; 1 - 0.05$$

Dans le premier cas, cette distribution convient bien car de nombreuses questions offrent des choix dichotomiques. Une réponse positive peut donc être considérée comme un succès. On prend alors la fréquence de réponses positives à une question comme estimateur de la proportion. Notons que, dans le cas de questions à connotation négative, un succès correspondra plutôt à une réponse négative.

Plusieurs questions offrent trois possibilités de réponse plutôt que deux. Nous les avons alors ramenées à deux possibilités en considérant les réponses strictement positives pour pouvoir utiliser la distribution binomiale. Nous avons donc un peu réduit l'information dont nous disposions. L'utilisation de la multinomiale est plus indiquée mais la construction de l'intervalle est beaucoup plus lourde. Cependant, il est bien clair que l'information reste accessible et peut être exploitée par ailleurs.

Dans le deuxième cas, il est également possible d'utiliser facilement la binomiale. En effet, on peut limiter l'échantillon aux individus susceptibles de progresser lors de la première séance (cela constitue l'effectif) et considérer qu'un progrès en quatrième séance par rapport à la

première correspond à un succès. On calcule alors une "fréquence de progrès".

Nous illustrons la façon dont on peut utiliser ces fréquences, et les intervalles de confiance qui y sont associés en présentant les résultats de l'évaluation dans la section 3.5.

3.3.2.5.2. L'analyse en composantes principales

Lorsqu'on possède une grosse masse de données c'est-à-dire un certain nombre d'individus et de variables, il devient très difficile d'interpréter l'information dont on dispose. L'analyse en composantes principales s'applique typiquement à ce genre de problèmes. En effet, il s'agit d'un outil descriptif qui a pour objectif de résumer l'information de manière à la rendre plus "maîtrisable".

Nous allons maintenant suivre les différentes étapes d'une analyse en composantes principales de manière à voir comment a lieu ce résumé de l'information.

Etant donné que nous avons utilisé le logiciel SAS pour faire les analyses en composantes principales que nous exposons dans les résultats de l'évaluation (cfr 3.5.), nous présentons cette théorie comme elle est réalisée avec SAS. Nous nous basons pour cela sur le cours de "Statistique appliquée" de Madame Noirhomme, sur le guide d'utilisation de SAS et sur "Analyse des données multi-dimensionnelles" de Bertier et Bouroche.

Au départ, on a n individus pour lesquels on a mesuré p variables quantitatives. Ces variables sont, dans notre cas, les questions posées. Les réponses sont codées et peuvent prendre 3 valeurs : 0, 0.5 et 1. Cette masse de données peut se présenter sous la forme d'un nuage de points dans un espace à p dimensions, chaque variable étant un des p axes. On commence par construire une matrice Y de dimensions n*p telle que la $i^{ème}$ ligne est le vecteur des mesures du $i^{ème}$ individu :

$$Y = \begin{vmatrix} y_1^1 & \dots & y_1^k & \dots & y_1^p \\ & & & & \\ y_i^1 & \dots & y_i^k & \dots & y_i^p \\ & & & & \\ y_n^1 & \dots & y_n^k & \dots & y_n^p \end{vmatrix}$$

yk_i est la $k^{ième}$ mesure du $i^{ème}$ individu.

Ensuite, on calcule la moyenne de chaque variable k:

$$\overline{y}^k = \frac{\sum_{i=1}^{n} y_i^k}{n}$$

ainsi que son écart-type :

$$s_k = \sqrt{\frac{\sum_{i=1}^{n} (y_i^k - \overline{y}^k)^2}{n-1}}$$

On peut ainsi centrer chaque observation yk_i en retranchant de celle-ci la moyenne de la $k^{ième}$ variable sur tous les individus et la standardiser en divisant l'observation centrée par l'écart-type de la variable. On obtient ainsi une nouvelle matrice X dont chaque élément xk_i est une observation centrée et standardisée :

$$x_i^k = \frac{y_i^k - \overline{y}^k}{s_k}$$

Centrer les différentes observations signifie amener les différents axes au centre du nuage de points.

La matrice des corrélations entre les variables

La matrice X nous permet de calculer très facilement la matrice des corrélations entre les différentes variables :

$$V = \frac{X'X}{n}$$

Chaque élément de cette matrice est la corrélation entre 2 variables. Ainsi, V_{kl} est la corrélation entre la $k^{ième}$ et la $l^{ième}$ colonne de X.

Cette matrice des corrélations permet déjà d'obtenir beaucoup d'informations. Ainsi, pour nos questions, nous pourrions voir quelles sont les questions auxquelles les individus répondent de façons fort semblables. Nous pourrions aussi mettre en évidence des questions fort corrélées négativement. Cela signifierait que lorsqu'on répond "oui" à la première question, on répond "non" à la seconde question et inversement.

Recherche de nouveaux axes

Partant de la matrice V, l'analyse en composantes principales va étudier les corrélations entre les différentes variables (que l'on appelle les axes initiaux) et déterminer des combinaisons linéaires de variables assez corrélées de manière à construire de nouveaux axes (appelés axes factoriels). On va prendre comme premier axe celui qui perd le moins d'information. Si on parle en terme de "nuage de points", le premier axe devra tenir compte au maximum de la dispersion du nuage. On l'obtient en maximisant l'expression : $Vu_1 = \lambda_1 u_1$.
Or, on a $\lambda_1 = u_1'Vu_1$.
Le premier axe sera donc le vecteur propre associé à la plus grande valeur propre.
De même, le second axe sera obtenu en prenant le vecteur propre associé à la seconde plus grande valeur propre λ_2,...
Pour chaque axe ainsi obtenu, on peut avoir une idée du pourcentage d'information dont il tient compte : il s'agit de

$$\frac{\lambda_k}{\sum_{i=1}^{p} \lambda_i}$$

On appelle ce nombre le pourcentage d'inertie expliquée par l'axe k. Généralement, on espère pouvoir se limiter à 2 ou 3 axes factoriels qui résumeraient bien l'information de départ. On calcule donc :

$$\frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$$

qui nous donne un pourcentage d'inertie expliqué par les axes 1 à q. Il faut vérifier si ce pourcentage est bien suffisant pour se limiter à q axes.

Interprétation des nouveaux axes

Une fois que l'on a trouvé les différents axes u_i et que l'on a décidé du nombre d'axes factoriels à garder, il reste à interpréter ceux-ci. C'est la principale difficulté de l'analyse en composantes principales. Ainsi, si nous prenons l'exemple du questionnaire, nous avons fait plusieurs analyses en composantes principales mais nous avons dû en abandonner un certain nombre car nous étions incapables de donner une interprétation (même floue) aux axes que nous avions trouvés! On ne pouvait alors rien en tirer d'intéressant...

Représentation des individus sur les nouveaux axes

Si on parvient à interpréter les axes trouvés, on peut alors représenter les différents individus dans le nouveau système d'axes. Dans l'ancien système d'axes, un individu était représenté par un vecteur ligne de la matrice X : $x_i = (x^1{}_i,...,x^p{}_i)$.
Pour obtenir sa représentation sur les nouveaux axes, il faut prendre sa projection sur ces nouveaux axes : $(x_i.u_1,...,x_i.u_q)$.

Corrélation entre les anciennes et les nouvelles variables.

Une ancienne variable k est représentée par un vecteur colonne de la matrice X :

$$x^k = \begin{vmatrix} x_1^k \\ \cdot \\ \cdot \\ \cdot \\ x_n^k \end{vmatrix}$$

Un nouvel axe est obtenu comme suit :

$$a_j = \begin{vmatrix} x_1 \cdot u_j \\ \\ \\ x_n \cdot u_j \end{vmatrix}$$

On peut trouver la corrélation entre les anciennes variables et les nouvelles. On parle alors de la corrélation variables-facteurs qui s'obtient comme suit:

$$\text{cor}\,(x^k, a^l) = \sqrt{\lambda_l} \cdot u_l^k$$

Cette corrélation est le principal élément qui nous aide à interpréter les nouveaux axes. Cependant, malgré cette information, il reste très souvent malaisé de donner une interprétation claire aux nouveaux axes.

Ainsi, si nous remarquons que deux questions sont très fortement corrélées avec l'axe qu'on veut interpréter, que l'une l'est positivement et l'autre négativement, nous disposons ainsi de beaucoup de renseignements pour parvenir à une interprétation de l' axe. Mais, si nous ne trouvons pas quel élément oppose ces deux questions, nous resterons incapables d'interpréter l'axe.

Notons que la représentation des individus sur les nouveaux axes peut également nous aider dans l'interprétation si nous connaissons suffisamment les individus pour savoir quelles sont leurs ressemblances et leurs différences...

3.3.2.5.3. Le traitement des données pairées

Nous présentons cette partie en nous basant sur le cours d'"Introduction à la mathématique statistique" de Madame Noirhomme ainsi que sur ses notes concernant "Les données pairées".

Lorsqu'on désire voir si deux traitements sont différents, on applique chacun de ces traitements à un échantillon et il suffit alors de tester l'égalité des moyennes des deux populations.

Cependant, lorsque les échantillons sont dépendants ou même constitué des mêmes individus, il n'est plus possible de procéder de la même façon.

Supposons que les deux traitements soient appliqués à la même personne, on a :

	traitement 1	traitement 2	différence
ind_1	$X_{1,1}$	$X_{1,2}$	$d_1 = X_{1,1} - X_{1,2}$
.	.	.	.
.	.	.	.
.	.	.	.
ind_n	$X_{n,1}$	$X_{n,2}$	$d_n = X_{n,1} - X_{n,2}$

On peut calculer la différence entre les deux traitements et on obtient ainsi des d_i que l'on peut considérer comme nos observations.

Nous pouvons calculer : $\partial = \sum d_i / n$, l'estimateur de la moyenne des différences;

et $$s_d^2 = \frac{\sum (d_i - \bar{d})^2}{n-1}$$,

si l'échantillon est petit et extrait d'une population normale.

On peut alors estimer la moyenne de la différence (**E**d) avec un niveau d'incertitude de α en calculant l'intervalle de confiance suivant :

$$Pr[\partial^{\wedge} - Q_{t_{n-1}}(1-\alpha/2)* s_d/\sqrt{n} \leq \partial \leq \partial^{\wedge} + Q_{t_{n-1}}(1-\alpha/2)* s_d/\sqrt{n}] = 1-\alpha$$

Il est nécessaire de consulter la table de la distribution t de Student pour obtenir les limites inférieures et supérieures de l'intervalle de confiance. Les entrées de la table sont $1-\alpha/2$ et n-1. Elle est reprise dans l'annexe 7.

Nous pourrions utiliser le traitement des données pairées afin de comparer la compréhension des différents éléments de l'interface lors de la première séance et de la quatrième. Ceci nous permettrait de voir s'il y a une différence significative entre ces deux moments.

3.3.2.5.4. Les méthodes non-paramétriques

Il arrive qu'on ne connaisse pas la distribution de la population qu'on désire étudier. Dans ce cas, on ne peut pas utiliser des méthodes statistiques qui se basent sur une loi de distribution particulière.

Il existe des méthodes qui ne tiennent pas compte de la distribution. On les appelle les méthodes non-paramétriques. Celles-ci ne font pas d'hypothèse sur la population étudiée. Elles peuvent être utilisées pour de petits échantillons. Ces méthodes sont assez nombreuses et peuvent être intéressantes.

On peut, par exemple, estimer la médiane de la population plutôt que sa moyenne car il n'est pas nécessaire de connaître la forme de la distribution de la population. La médiane est un nombre tel qu'il y a autant de chances (50%) pour qu'une observation se trouve à sa gauche qu'à sa droite.

Nous présentons brièvement la manière dont on peut estimer la médiane en nous basant sur le cours d' "Introduction à la statistique mathématique" donné par Madame Noirhomme.

Pour commencer, il est nécessaire de classer les informations de l'échantillon comme suit :

$(\underline{x}1,..., \underline{x}n)$ tel que $\underline{x}1 \leq \underline{x}2 \leq ... \leq \underline{x}n$

avec $\underline{x}_i$ = la i ème observation de l'échantillon ordonné;
n = l'effectif de l'échantillon.

Nous pouvons ensuite calculer l'estimateur de la médiane de la population en prenant la médiane de l'échantillon :

$\hat{med} = \underline{x}k+1$ si n = 2k+1
$= 1/2(\underline{x}k+\underline{x}k+1)$ si n = 2k

Il serait également intéressant de connaître un intervalle de confiance pour la médiane. Comme la probabilité qu'une observation se trouve à gauche est égale à 1/2, on peut dire que le nombre d'observations à gauche de la médiane a une distribution binomiale de signature (n,1/2). On a évidemment la même chose à droite de la médiane. On peut écrire:

$$Pr[\underline{x}_r < med < \underline{x}_s] = \sum_{i=r}^{s-1} C_n^i (1/2)^n$$

en prenant r<s.

On choisit souvent s=n-r+1 afin de faciliter les calculs. Il suffit alors de chercher r tel que :

$$(1/2)^n \sum_{i=r}^{n-r} C_n^i \geq 1-\alpha$$

pour obtenir les limites inférieure et supérieure de l'intervalle de confiance.

Nous appliquons cette méthode pour pouvoir répondre à certains objectifs de l'évaluation.

3.3.3. LES TRACES D'UTILISATION

3.3.3.1. Définition

Les traces d'utilisation sont constituées par la suite chronologique des manipulations effectuées par une personne handicapée lors d'une session d'utilisation d'un logiciel.

3.3.3.2. Les traces enregistrées par le programme Comptes

Après chaque séance, le programme Comptes enregistre sous la forme d'un fichier de type texte le numéro de la séance, le nom de l'utilisateur, les paramètres choisis ainsi que les actions menées au cours de l'utilisation.

Dans le cas particulier du programme Comptes, nous préférons utiliser le terme "action" plutôt que celui de "manipulation" car en fait, toutes les manipulations ne sont pas enregistrées. Les mouvements de la souris, par exemple, sont des manipulations mais ils ne sont pas mémorisés. Il n'est donc pas possible de reconstruire parfaitement une session d'utilisation à partir des traces. Seules les actions qui correspondent à un "clic" de la souris sur un objet interactif ou une frappe sur la touche "return" sont reprises dans les traces d'utilisation du programme Comptes.

3.3.3.3. Utilisation de l'outil pour répondre aux objectifs de l'évaluation

C'est essentiellement pour évaluer les interfaces et la durée d'apprentissage du programme que l'outil "traces" pourra être utilisé. Mais celui-ci ne nous permettra pas de juger de l'intérêt ou de l'utilité du logiciel.

Certaines "erreurs" de manipulation peuvent apparaître dans les traces. Ainsi, lorsque l'utilisateur quitte une fonction sans la mener à son terme, infirme une demande ou encore désire recommencer la dernière action effectuée, cela traduit généralement une difficulté. Dès lors, en dépouillant toutes les traces d'un utilisateur ou d'un ensemble d'utilisateurs, nous pourrons découvrir que la fréquence de certaines erreurs de manipulations est assez élevée et donc supposer qu'un écran

pose problème, qu'un terme est mal compris ou ambigu, qu'une icône est mal dessinée, que le déroulement des écrans doit être revu, ...

De plus, la comparaison de ces fréquences d'erreurs de séance en séance pour un même utilisateur pourrait peut-être nous aider à mesurer des progrès réalisés par la personne handicapée.

Très souvent, les traces ne nous permettent pas à elles seules de tirer des conclusions sur l'évaluation de l'interface du programme. Elles ne constituent que des indices à confronter aux résultats fournis par les autres outils à savoir le questionnaire mais aussi l'avis de l'éducateur et de l'utilisateur.

3.3.3.4. Méthode d'exploitation des traces

Afin d'exploiter systématiquement les traces d'utilisation, le programme Comptes a été découpé en 26 états. Un état correspond à un écran ou à une partie d'écran (s'il s'agit d'une boîte de dialogue) dans lequel plusieurs actions possibles sont proposées. Certaines actions, si elles sont déclenchées, feront passer le programme dans un autre état. Ce passage d'un état à un autre est une transition. Les transitions possibles entre états peuvent donc s'écrire dans une matrice carrée qui comporte autant de lignes que d'états différents: il s'agit de la matrice des transitions possibles.

Dans la matrice des transitions possibles, la cellule ij contient :
- 0 s'il n'est pas possible dans l'état i de passer directement à l'état j;
- 1 si, dans l'état i, on ne peut passer qu'à l'état j;
- x si, dans l'état i, il est possible de passer directement à l'état j mais que d'autres états sont également accessibles à partir de l'état i.

Le lecteur plus intéressé par cet outil d'évaluation pourra se référer au mémoire de Gianni Strappazzon et Marc Déplechin : "Un logiciel de gestion de comptes pour personnes handicapées mentales: développement et évaluation".

Remarquons que, d'un état donné, on ne peut accéder directement (c'est-à-dire en une seule transition) qu'à un ensemble limité et déterminé d'autres états.

Sur base de cette matrice, nous pensons qu'il est possible de faire correspondre à chaque trace sa matrice des transitions effectives dont chaque cellule ij contient le nombre de transitions effectivement franchies entre l'état i et l'état j. On pourrait également construire une seule

matrice à partir de toutes les traces d'un utilisateur ou même les traces d'un ensemble d'utilisateurs et obtenir ainsi une information plus "agrégée".

Certaines "erreurs" de manipulations correspondent à des transitions critiques. Celles-ci pourront être facilement localisées dans la matrice de transitions effectives et attireront particulièrement notre attention.

Il est évidemment toujours possible de construire la matrice des transitions effectives à partir d'une trace, mais on perd un peu d'information au cours de cette opération. En effet, on ne connaît plus la chronologie des actions c'est-à-dire l'ordre dans lequel l'individu est passé d'un état à un autre. De plus, certaines actions effectuées par l'utilisateur ne correspondent pas à un changement d'état et n'apparaissent dès lors plus dans la matrice des transitions.

Si les questionnaires et les traces d'utilisation permettent de recueillir de l'information pour l'évaluation, il est tout aussi vrai que les entrevues avec les éducateurs et les psychologues ainsi que les contacts directs avec les utilisateurs constituent eux aussi de véritables outils d'évaluation. Les rencontres et discussions avec les personnes peuvent, en effet, nous apporter de nombreuses informations qui seraient difficilement révélées par un questionnaire ou une trace de manipulation à cause de leur caractère formel.

3.3.4. LES CONTACTS AVEC LES UTILISATEURS

Dans tout projet informatique, il est nécessaire que l'utilisateur participe aux différentes phases du cycle de vie du projet. En particulier dans la phase d'évaluation, le rôle de l'utilisateur est primordial. En effet, le projet lui est adressé et donc, c'est lui le mieux placé pour vérifier s'il correspond bien à ses objectifs.

C'est la raison pour laquelle nous avons commencé notre mémoire par un stage dans une institution où nous avons partagé quelques jours la vie de ces personnes. Nous avons réalisé plusieurs activités avec celles-ci de manière à mieux les connaître et nous avons participé à plusieurs utilisations du programme.

3.3.5. LES ENTREVUES AVEC LES EDUCATEURS ET PSYCHOLOGUES

Dans le cadre d'un projet pour personnes handicapées mentales, l'évaluation par les utilisateurs eux-mêmes n'est pas simple et nécessite une approche particulière. Les éducateurs et psychologues deviennent alors des tiers acteurs indispensables.

D'une part, ils ont accumulé des connaissances théoriques et pratiques concernant les capacités, les aptitudes, les acquis des personnes handicapées mentales. En ce sens, ils peuvent émettre un avis tout à fait éclairé sur le programme, sur son utilité,...

D'autre part, ils connaissent suffisamment les personnes handicapées mentales pour comprendre et interpréter leur langage, leurs comportements,... et donc pour en être des interprètes fidèles auprès des personnes chargées de l'évaluation du logiciel.

3.4. LES PROBLEMES RENCONTRES LORS DE LA PHASE D'EXPERIMENTATION

Lors de la phase d'expérimentation, nous avons rencontré deux grands types de problèmes.

Premièrement, nous n'avons pas pu expérimenter le logiciel avec l'échantillon de 50 individus qui avait été initialement prévu.

En effet, certains éducateurs ne se sont pas bien rendus compte de la quantité de travail qui leur était demandée et nous ont donc fait des promesses qu'ils n'ont pas pu tenir. Rappelons que nous demandions à l'éducateur de se familiariser avec l'ordinateur Amiga et avec la manipulation du programme Comptes et de choisir des utilisateurs potentiels. Pour chacun de ceux-ci, l'éducateur devait créer les conditions pour qu'une expérimentation soit possible, remplir la partie psychologique du questionnaire, assister et guider les utilisateurs au minimum au cours de 5 séances d'utilisation du programme, remplir la partie ergonomique du questionnaire après chacune de ces séances. Enfin, une fois l'expérimentation terminée, il devait remplir une nouvelle fois la partie psychologique du questionnaire, donner ses commentaires sur les différents écrans et aider la personne handicapée à répondre au questionnaire destiné à l'utilisateur.

Des problèmes matériels sont aussi survenus. Différentes institutions intéressées par l'expérimentation n'ont pas pu acquérir ou disposer d'un ordinateur Amiga équipé d'une mémoire centrale d'un Mégabyte et de 2 lecteurs de disquettes. En outre, une institution belge a été victime d'un vol d'ordinateur en cours d'expérimentation.

Suite à toutes ces difficultés, nous n'avons pu récolter que 20 questionnaires remplis pour 4 séances et 6 questionnaires complétés pour une ou deux séances. Nous n'avons reçu aucun résultat des institutions françaises et très peu des institutions suisses.

Deuxièmement, un certain nombre d'éducateurs ont connu quelques désagréments durant la phase d'expérimentation.

La plupart des éducateurs n'avaient jamais été en contact avec un ordinateur auparavant. Même si le maniement des disquettes, l'exécution du programme Paramètres et du programme Comptes nous semblent faciles, nous nous sommes rendus compte que ce n'était pas toujours le cas pour eux. Nous avons donc pris conscience de la nécessité de mettre à leur disposition un mode d'emploi technique. Nous en avons donc rédigé un qui est présent dans l'annexe 4.

Par ailleurs, le programme contenait quelques erreurs qui, lorsqu'elles se présentaient, avaient pour effet d'interrompre complètement la session en cours : il était alors nécessaire de "relancer" l'ordinateur. Ces erreurs n'ont pas eu lieu fréquemment mais elles pouvaient provoquer chez l'éducateur et surtout chez l'utilisateur un regrettable sentiment de panique et d'impuissance. Nous avons donc été appelés à réagir le plus rapidement possible et à corriger ces erreurs.

3.5. LES RESULTATS DE L'EVALUATION

L'échantillon dont nous disposons comporte 26 individus dont 20 ont utilisé le programme lors de 4 séances et 6 lors d'une ou de deux séances. Le plus souvent, nous nous basons sur les 20 individus à 4 séances pour tirer des conclusions.

La taille de notre échantillon est donc assez petite et les conclusions que nous tirons indiquent de grandes tendances au sein de celui-ci. Nous ne prétendons pas les étendre à l'ensemble de la population correspondant aux prérequis fixés pour utiliser le programme.

Cet élément étant bien clarifié, nous pouvons maintenant, et ce, pour chacun des objectifs de l'évaluation, citer les outils que nous avons utilisés, la manière dont nous les avons exploités et les résultats que nous avons ainsi obtenus.

Lorsque nous examinons les réponses au questionnaire, nous calculons souvent des fréquences et des intervalles de confiance mais nous ne les citons pas chaque fois afin de ne pas surcharger l'exposé. Tout lecteur qui désire disposer d'une plus grande précision dans les résultats peut les consulter en annexe 6.

De plus, nous faisons référence aux questions en indiquant leurs numéros. Le lecteur pourra ainsi retrouver leur intitulé exact en consultant le questionnaire présent en annexe 1.

L' ensemble des questionnaires remplis et des traces de manipulation fait l'objet d'un second volume d'annexes disponible auprès de Madame Noirhomme.

L'UTILITE DU PROGRAMME DE TENUE DE COMPTES EN TANT QU'OUTIL D'AUTONOMIE

1) Le logiciel permet-il à une personne handicapée mentale de tenir ses comptes de manière autonome ?

Il est impossible de donner à cette question une réponse valable pour l'ensemble de la population handicapée.

Aucune personne handicapée mentale n'a pu utiliser le programme Comptes sans aucune aide ni aucune explication préalable dès la première séance d'expérimentation.

Dans différentes institutions, les éducateurs nous ont certifié que certaines personnes handicapées mentales avaient utilisé le programme pour tenir effectivement leurs comptes de manière autonome.

D'autres personnes handicapées mentales n'ont pas été capables de mener à bien l'activité de tenue de comptes de manière autonome après 5 séances. Néanmoins, les éducateurs estiment qu'après une vingtaine de séances d'utilisation, elles pourraient le faire. Pour ces personnes, on constate généralement très peu de problèmes liés à la manipulation de l'ordinateur, à la manipulation de la souris, au passage d'un écran à un autre,...

Un troisième groupe d'utilisateurs du logiciel ne pourra sans doute jamais prétendre pouvoir tenir des comptes de manière autonome. L'utilisation du logiciel pour ces personnes peut néanmoins être fort bénéfique comme nous allons le voir.

2) Quelle est l'utilité du logiciel Comptes ?

Le logiciel Comptes facilite beaucoup l'activité de tenue de comptes de la personne handicapée mentale qui enregistre simplement ses recettes, ses dépenses et l'état de son porte-monnaie mais qui ne doit rien calculer. En effet, la somme des recettes, celle des dépenses et la vérification de l'équilibre des comptes se font automatiquement avec le logiciel.

Parmi les personnes qui tenaient déjà leurs comptes avant l'expérimentation, plusieurs inscrivaient leurs recettes et dépenses dans un cahier de comptes et calculaient le solde en fin de semaine ou de mois. Ce calcul se faisait avec une machine à calculer et avec l'aide de l'éducateur. La plupart préfèrent l'utilisation de l'ordinateur et du logiciel à celle de la machine à calculer parce que "c'est plus facile! ". D'autres n'expriment aucune préférence entre l'ancienne et la nouvelle "méthode de tenue des comptes".

La facilité d'utilisation du logiciel est donc la conséquence du fait que l'ordinateur effectue tous les calculs mais surtout du fait qu'il "gère" lui-même les opérations qui doivent être effectuées pour tenir les comptes. Il "guide" beaucoup l'utilisateur en lui suggérant d'enregistrer ses recettes, ses dépenses et le montant contenu dans son porte-monnaie, en s'occupant de vérifier s'il y a équilibre des comptes,... Par exemple, l'utilisateur ne doit pas nécessairement savoir qu'il faut effectuer la somme des recettes, la somme des dépenses, soustraire la somme des dépenses de celle des recettes et comparer le résultat obtenu à l'état du porte-monnaie pour vérifier s'il y a équilibre des comptes. Tout cela se fait de façon implicite grâce au logiciel. En outre, le programme mémorise le solde de la session précédente.

Par conséquent, certaines personnes handicapées mentales peuvent travailler seules après quelques séances et l'éducateur peut ainsi gagner du temps.

Les inconvénients relevés sont les suivants :

- le fait que l'ordinateur ne peut jamais vérifier que la personne handicapée mentale ne se trompe pas lorsqu'elle enregistre la somme

contenue dans son porte-monnaie alors qu'un éducateur peut exercer un meilleur contrôle;
- les institutions intéressées devront consentir à l'achat de l'ordinateur Commodore Amiga mais le prix de celui-ci est abordable et il peut servir à d'autres activités.

3) La personne handicapée mentale éprouve-t-elle un certain plaisir à tenir ses comptes avec le logiciel?

Selon les éducateurs, l'utilisation du logiciel est fortement liée à un certain plaisir pour la personne handicapée mentale. L'avis des utilisateurs le confirme d'ailleurs. En effet, tous ont dit que cette activité de tenue des comptes les amusait. Plusieurs personnes étaient déçues de voir l'expérimentation se terminer.

Ce plaisir semble lié au logiciel lui-même : les couleurs, les dessins, le son séduisent beaucoup de personnes handicapées mentales. Mais il est également très lié au fait que l'utilisation du logiciel implique de travailler sur ordinateur. Ainsi, des personnes qui n'ont pas pris part à l'expérimentation et qui n'ont pas vu le logiciel enviaient parfois celles qui ont pu utiliser l'ordinateur. En effet, le fait de "travailler avec l'ordinateur" donne parfois l'impression aux personnes handicapées mentales qu'elles sont considérées comme des adultes. De plus, lorsqu'elles peuvent travailler seules, ce sentiment de revalorisation est renforcé.

Certaines ont le sentiment qu'elles contrôlent la machine et leur confiance en elles augmente.

Elles apprécient également la neutralité de la machine par rapport à un éducateur. Elles peuvent faire une activité sans être aidées et sans que quelqu'un ne leur fasse de remarques. De plus, la relation avec l'argent est particulière et il est préférable que les personnes handicapées n'aient pas l'impression que l'éducateur est un intrus qui s'occupe de ce qui ne le regarde pas. L'ordinateur n'est pas considéré de cette manière.

Ces différents éléments contribuent au plaisir que peut ressentir une personne handicapée mentale à utiliser le logiciel.

L'ordinateur semble donc fasciner beaucoup de personnes handicapées mentales mais il est bien évident qu'il y a parmi celles-ci aussi des personnes que l'informatique et les ordinateurs laissent indifférentes. Lors de nos contacts avec les personnes handicapées

mentales, nous avons rencontré une personne qui visiblement s'intéressait beaucoup à l'informatique et qui nous a parlé de virus et une autre qui, par contre, n'a cessé de critiquer avec ardeur les ordinateurs...

4) Le logiciel peut-il inciter des personnes handicapées mentales à tenir leurs comptes ?

Il semble que l'on puisse répondre positivement à cette question. En effet, quelques personnes qui ne tenaient pas leurs comptes avant l'expérimentation ont pris conscience de l'importance de l'argent et ont décidé après l'expérimentation de continuer à tenir leurs comptes sur un cahier de comptes avec l'aide de l'éducateur. Pour ces personnes, le logiciel a véritablement joué un rôle de "catalyseur". Un résultat comme celui-ci est très positif et encourageant.

L'UTILITE DU PROGRAMME EN TANT QU'OUTIL D'APPRENTISSAGE

5) Le logiciel permet-il d'apprendre de nouveaux termes ?

Nous abordons cette question en la subdivisant en deux parties:
- y a-t-il apprentissage dans le contexte particulier du programme ?
- y a-t-il apprentissage grâce au logiciel même hors de ce contexte ?

En effet, un individu pourrait apprendre et comprendre un nouveau terme lorsqu'il utilise le logiciel Comptes mais ne pas le comprendre en dehors de ce contexte. Il y aurait donc eu un apprentissage du premier type mais pas du second. Le passage de l'un à l'autre demande un effort intellectuel et peut prendre un certain temps. Il pourrait même ne jamais se faire pour certains.

Notons que nous ne parlons pas seulement de l'apprentissage de nouveaux termes mais aussi de l'amélioration de la compréhension de certains termes. Nous considérons donc, par exemple, que le passage d'une compréhension nulle à une compréhension vague d'un terme est un progrès.

Pour répondre à la première partie de la question, nous nous baserons sur trois questions du questionnaire ergonomique.

La première (E18) concerne la compréhension des termes utilisés dans le menu, c'est-à-dire "recette", "dépense", "état du porte-monnaie" et

"fin". Il s'agit donc de "termes-clé" dans une tenue de comptes. Parmi les 10 individus qui ne connaissaient pas parfaitement ces termes lors de la première séance, 6 ont progressé à la quatrième séance (ou avant la quatrième séance). Cela constitue donc une fréquence de progrès de 60%, ce qui apparemment constitue un bon score, le processus d'apprentissage étant généralement assez lent chez les personnes handicapées mentales. Si on calcule l'intervalle de confiance pour cette fréquence avec un niveau d'incertitude 0.05, on obtient $0.27 \leq p \leq 0.88$, ce résultat est moins encourageant. Cependant, cet intervalle ne recouvre pas 0 et on peut conclure qu'il y a apprentissage des termes par certains individus et que le logiciel permet donc l'apprentissage du moins dans le contexte du programme.

Les deux autres questions concernent le fait que la personne se base sur le texte dans les écrans de saisie du jour (E26) et du poste (E23) d'une recette ou dépense. Elles sont moins nettement liées à l'apprentissage des termes mais peuvent constituer des indices car si une personne se base plus sur le texte en fin d'expérimentation qu'en début, cela peut se justifier par une meilleure compréhension du texte et donc, par un apprentissage de termes. En fait, on constate qu'aucun individu ne se base plus sur le texte lors de la quatrième séance que lors de la première. Cela peut peut-être s'expliquer par le fait que les termes utilisés dans ces 2 écrans sont sans doute plus courants que des mots comme "recette" ou "état du porte-monnaie", et donc que, s'il n'y a pas eu apprentissage de ces termes avant l'expérimentation, il y a très peu de chances pour qu'il se fasse en 4 séances. Il s'agit en effet de mots tels que lundi, mardi, transports, nourriture,...

Nous vérifions maintenant si des termes sont appris et compris même en dehors du contexte du programme. Deux questions issues du questionnaire psychologique nous permettent de nous faire une idée à ce propos : elles concernent la compréhension des termes "recette" (P32) et "dépense" (P33) avant l'expérimentation et après celle-ci. Parmi les 13 personnes qui ne connaissent pas parfaitement le terme "recette" avant l'expérimentation, 4 le comprennent mieux après l'expérimentation. Cela nous donne une fréquence d'apprentissage de 31% et l'intervalle de confiance $0.09 \leq p \leq 0.62$ avec un niveau d'incertitude 0.05. Il y a donc apprentissage du mot "recette" même en dehors du programme. En procédant de la même façon pour le mot "dépense", on constate que seuls

6 individus ne connaissent pas parfaitement le terme avant l'expérimentation. Il semble donc mieux compris que "recette". Après l'expérimentation, 3 individus ont progressé: la fréquence de progrès est donc de 50 % et l'intervalle de confiance est $0.12 \leq p \leq 0.87$. De nouveau, cet intervalle ne recouvre pas 0 mais on peut constater qu'il est vraiment très étendu. En effet, l'effectif de l'échantillon est de 6, ce qui est très peu...

Il apparaît donc que le logiciel permet d'apprendre de nouveaux termes non seulement dans le contexte de son utilisation, ce qui constitue déjà un point positif mais également en dehors d'un contexte particulier, ce qui est évidemment encore plus intéressant!

6) Le logiciel permet-il de comprendre de nouveaux concepts?

Nous faisons ici la même remarque que pour l'apprentissage de termes : nous considérons qu'un progrès correspond à une compréhension d'un concept meilleure qu'auparavant mais n'implique pas nécessairement une compréhension parfaite du concept en fin d'expérimentation.

Nous ne disposons pas de beaucoup d'éléments pour répondre à cette question. En effet, aucun éducateur n'a révélé de progrès dans la compréhension de concepts.

Pour certains, cependant, il semble que le logiciel permettrait de mieux comprendre les concepts de "recette", "dépense", "équilibre des comptes",... mais que cela ne pourrait se faire qu'à long terme. Sur une période d'un mois, cela paraît impossible.

De plus, notre échantillon comprend presque uniquement des personnes qui comprenaient déjà les concepts "recette" et "dépense" même si elles ne connaissaient pas les termes et si elles n'y prêtaient pas encore attention. En fait, il semble que les éducateurs effectuent une "sélection" parmi les personnes handicapées et choisissent d'expérimenter le logiciel avec des personnes qui ont déjà quelques notions relatives à l'argent.

Nous ne pouvons donc pas donner de réponse nette à cette question. Pour confirmer l'avis des éducateurs, il faudrait mener une expérimentation à plus long terme.

7) Le logiciel permet-il des apprentissages au niveau des acquis de la personne handicapée mentale ?

En ce qui concerne les acquis en lecture, écriture et calcul, nous n'avons pas posé de questions "avant" et "après" l'expérimentation car il est apparu clairement, suite à des discussions avec des psychologues, qu'on ne pouvait pas constater de changements significatifs en 4 séances c'est-à-dire en 4 semaines mais qu'il faudrait des mois si pas des années pour que de tels changements surviennent. Ce temps dépend bien sûr de la personne handicapée mentale. Certaines ne pourront d'ailleurs quasiment pas progresser dans ce domaine.

Cependant, on peut espérer un apprentissage à long terme. En effet, une logopède nous a appris qu'après 4 séances d'utilisation du logiciel, une personne handicapée mentale qui ne sait lire que des lettres séparément et qui ne sait pas les mettre ensemble pour constituer un mot a été capable de reconnaître le mot "oui" dans toute une liste de mots de 3 lettres. En effet, lorsqu'une confirmation est demandée, le mot "oui" est toujours placé à gauche et à force de le voir, la personne sait maintenant le reconnaître même dans un autre contexte que celui du logiciel. On ne peut bien sûr pas dire que la personne a appris à lire mais elle reconnaît un mot et cela constitue une première étape en vue d'un apprentissage de la lecture.

8) Le logiciel peut-il entraîner des changements dans les comportements des individus vis-à-vis de l'argent?

Le logiciel, en provoquant une prise de conscience de l'importance de l'argent dans la vie quotidienne, pourrait amener la personne handicapée mentale à modifier ses comportements en rapport avec l'argent. Elle pourrait, par exemple, commencer à vérifier la monnaie qu'on lui rend à la caisse. Trois questions sont relatives à ce comportement.

La première (P40) concerne la monnaie rendue sur une somme de 100 FB. On constate que, parmi les 9 individus qui ne vérifiaient pas la monnaie avant l'expérimentation, un seul le fait après. La fréquence d'apprentissage de ce comportement est donc de 11 % et l'intervalle de confiance pour cette fréquence recouvre 0. Nous ne pouvons donc pas conclure qu'il y a apprentissage.

Pour une somme de 500 FB (P41), personne ne progresse et pour 1000 FB (P42), on observe un léger progrès. On ne peut donc pas parler d'un net apprentissage pour ce comportement. Cependant, il ne faut pas s'en étonner car nous-mêmes ne prenons pas toujours le temps de vérifier l'argent rendu par les caissières!

Il est nettement plus intéressant de savoir si une personne essaie de retenir les dépenses et recettes qu'elle a faites et de voir comment elle le fait. Trois moyens nous sont apparus :

- faire confiance à sa mémoire et essayer de tout retenir sans support (P43, P44, P45);
- noter sur un support papier ses recettes et dépenses (P46);
- garder ses tickets de caisse (P47).

Il apparaît qu'après l'expérimentation, deux changements significatifs ont eu lieu : il y a eu un effort de mémorisation et certains gardent désormais leurs tickets. Par contre, personne n'a commencé à utiliser un support papier en cours d'expérimentation. Seuls ceux qui l'employaient déjà ont continué à s'en servir.

Par ailleurs, les personnes ne déchiffrent pas mieux leurs tickets de caisse qu'auparavant (P27 à P31). Mais un individu, sans doute parce qu'il ne comprenait pas bien les différentes rubriques inscrites sur son ticket, a pris l'habitude de dessiner les articles qu'il avait achetés à côté de leur prix.

Outre ces changements de comportements dépistés par le questionnaire, d'autres habitudes ont également changé ou sont susceptibles de le faire. Ainsi, parmi les personnes handicapées mentales, il apparaît selon quelques éducateurs que certaines dépensent tout l'argent qu'elles reçoivent pour la semaine ou le mois en un laps de temps très court et se retrouvent ensuite sans argent jusqu'à leur prochaine recette. Certaines pourraient essayer de mieux répartir leurs dépenses. D'autres ont même envie d'épargner pour s'acheter une maison par exemple. Suite à l'utilisation de ce logiciel de tenue de comptes, on pourrait bien voir apparaître la volonté d'une véritable gestion de l'argent.

En tout cas, les éducateurs considèrent que le logiciel permet des changements de comportements très profitables. Ainsi, il semble que le logiciel pourrait pousser les personnes handicapées mentales à adopter des comportements et habitudes qu'elles garderaient même au-delà de l'utilisation du logiciel. Il permettrait donc de les préparer à une vie

autonome. En effet, certains individus de notre échantillon vivent en institutions et sont préparés à vivre seuls.

9) Le logiciel permet-il aux personnes handicapées mentales d'acquérir une meilleure notion des valeurs?

Le logiciel pourrait améliorer de façon indirecte la notion des valeurs des personnes handicapées mentales. En effet, il provoque souvent une prise de conscience de l'importance de l'argent et donc un plus grand intérêt pour celui-ci. Il est donc possible que les personnes handicapées mentales, étant plus attentives à tout ce qui touche à l'argent, commencent notamment à mieux connaître la valeur relative des pièces et des billets. L'expérimentation ne nous a pas permis d'établir cet élément puisqu'une seule question (P25) concernait justement cette connaissance et qu'aucun progrès ne s'est manifesté.

Par contre, un léger progrès a eu lieu dans la notion de l'état du porte-feuille (P36 à P39). De même, la notion du prix des choses s'est améliorée très légèrement pour des objets que les personnes achètent (P34) et beaucoup plus significativement pour les autres objets (P35).

Le logiciel peut intervenir également plus directement dans la notion des valeurs des personnes handicapées mentales qui l'utilisent. En effet, le thermomètre fluctue proportionnellement à la somme enregistrée. Il devrait donc permettre aux individus d'apprendre des notions comme "plus grand", "égal",... Cependant, il ne joue pas du tout ce rôle car il est mal compris comme nous le verrons un peu plus tard.

En fait, le logiciel ne joue qu'un rôle indirect sur la notion des valeurs des individus mais ce rôle existe.

L'INTERFACE ET LES FONCTIONNALITES DU LOGICIEL

10) Les fonctionnalités offertes par le logiciel sont-elles suffisantes et satisfaisantes ?

Il semble difficile de poser des questions fermées à ce sujet. Nous avons donc discuté avec les éducateurs pour recueillir leurs critiques et leurs propositions.

Il apparaît qu'une possibilité de corriger l'état du porte-monnaie s'impose car une éventuelle erreur se répercute de séance en séance puisque l'état du porte-monnaie d'une session devient une recette lors de la session suivante. Il est donc primordial que cette somme soit exacte. Le programme impose déjà une confirmation de la somme enregistrée mais il est arrivé qu'une personne retrouve de l'argent dans ses poches en cours de session et qu'elle soit alors incapable de remettre le montant à sa juste valeur.

Dans les institutions qui n'utilisent pas le logiciel en tant qu'outil d'autonomie mais seulement en tant qu'outil d'apprentissage, il serait intéressant que l'état du porte-monnaie d'une séance ne soit pas considéré comme une recette pour la suivante mais qu'il soit possible de repartir à zéro chaque fois. Ceci pourrait faire l'objet d'un paramètre.

La tenue des comptes sur un cycle d'un mois nous a été réclamée pratiquement partout. Cela semble vraiment être une importante lacune du programme car une tenue des comptes par mois correspond beaucoup mieux qu'un cycle d'une semaine ou d'un jour à la réalité de nombreuses personnes handicapées mentales et leur faciliterait la tâche car cela leur permettrait de ne pas devoir subdiviser leur tenue de comptes en 4 ou 5 semaines.

Enfin, et ici, nous "sortons" des fonctionnalités du logiciel qui a pour but de tenir les comptes, une véritable gestion du budget est attendue avec beaucoup d'impatience et devrait compléter la tenue des comptes. Beaucoup d'individus aimeraient pouvoir faire des prévisions, épargner,... Cela ne constitue pas une critique du logiciel mais il nous semble important de signaler qu'une forte demande existe pour un programme de gestion de budget.

Nous avons également remarqué que personne n'a tenu ses comptes sur un cycle d'un jour. Cette possibilité est peut-être inutile. En fait, elle est très contraignante puisqu'elle exige que la personne utilise l'ordinateur tous les jours où elle fait des recettes ou dépenses.

11) L'interface réalisée dans le logiciel convient-elle bien aux personnes handicapées mentales ou devrait-elle être améliorée?

Cette question est en fait très vaste.

On peut se demander si l'utilisation du clavier ou celle de la souris posent problème pour les personnes handicapées mentales.

D'après les discussions que nous avons pu avoir avec les éducateurs, il semble que beaucoup de personnes utilisent très facilement la souris. Plusieurs éducateurs ont d'ailleurs été étonnés par ce fait.

Quelques questions (E10, E20, E45) concernent les imprécisions que pourrait commettre la personne handicapée mentale en utilisant la souris. Elles tendent à confirmer le fait qu'il n'y a pas lieu de s'inquiéter sur ce point.

En ce qui concerne la manipulation de la souris, nous avons appliqué la méthode de l'analyse en composantes principales. Celle-ci contenait 8 questions que nous avons choisies en puisant une question dans chaque rubrique du questionnaire psychologique :

P4 : La personne lit un texte simple
P9 : La personne écrit des phrases courtes
P18 : La personne multiplie
P25 : La personne connaît la valeur relative des pièces et des billets
P26 : La personne est capable de vérifier si la somme est exacte
P28 : La personne peut distinguer les différentes rubriques d'un ticket de caisse
P35 : La personne a une notion du prix des choses pour des objets qu'elle n'achète pas
P42 : La personne paie et vérifie effectivement la monnaie rendue sur 1000 FB

et 3 questions ergonomiques fort liées à la manipulation de la souris :

E10 : Il arrive que la personne clique sur un autre billet (pièce) que celui désiré
E20 : La personne clique sur "Fin" par inadvertance
E46 : La personne clique sur des objets non-interactifs.

Nous invitons le lecteur à consulter l'annexe 8 qui présente les résultats de cette analyse. Ils montrent que deux axes permettent d'expliquer 65 % de l'information. Nous avons interprété le premier axe

comme étant un indicateur d'un certain "niveau" de la personne car il est fort lié à toutes les questions psychologiques et très faiblement aux trois autres questions. Les individus qui rencontrent beaucoup de difficultés se trouvent bien à gauche par rapport à cet axe alors que ceux dont les acquis et les capacités sont plus nombreux se trouvent plutôt à l'extrême droite.

Le second axe est très fortement lié positivement aux questions E10 et E20 et on pourrait considérer qu'il s'agit donc de la facilité de manipulation de la souris.

On constate alors que le niveau de la personne et sa capacité à manipuler la souris semblent tout à fait indépendants. Des personnes qui ont beaucoup d'acquis et de notions ne sont pas nécessairement plus agiles que les autres pour manipuler la souris et inversement. Cela confirme bien la réalité puisqu'il apparaît que beaucoup de personnes handicapées mentales manipulent la souris avec autant d'aisance que n'importe qui, sauf évidemment si elles ont un handicap moteur associé.

Cependant, nous devons nuancer nos conclusions car ce deuxième axe n'est pas uniquement lié à ces deux questions. D'autres interfèrent et pourraient fausser un peu les résultats. L'interprétation des axes obtenus grâce à l'analyse en composantes principales est parfois difficile.

Selon les éducateurs, la manipulation du clavier pose parfois des problèmes. Nous avons cherché confirmation dans le questionnaire. Il en ressort que les personnes handicapées mentales éprouvent en effet plus de difficultés à utiliser le clavier que la souris. Les problèmes semblent légèrement plus importants pour la saisie des chiffres (E15) que pour celle du nom (E1). Un élément les perturbe beaucoup : le fait de devoir taper sur la touche "return" (E17). Des éducateurs nous ont appris que certaines personnes attendaient chaque fois quand elles avaient terminé de taper un montant ou leur nom, pensant que quelque chose se passerait automatiquement.

Nous aimerions maintenant tenter de déterminer si le texte, les dessins, les couleurs et le son sont bien compris par les personnes handicapées mentales.

Parmi les termes utilisés, il apparaît que certains ne sont pas satisfaisants car ils sont ambigus et parfois confondus par des individus.

Ainsi, les réponses aux questions nous apprennent que la différence entre les mots "Efface" et "Recommencer" (E2) de même qu'entre

"Quitter" et "Fini" (E3) est mal comprise par certaines personnes. Le terme "Fini" qui indique qu'on veut passer à un écran suivant n'est pas parfaitement maîtrisé (E13) et, selon les éducateurs, une confusion s'établit parfois entre les mots "Fini" et "Fin", ce dernier correspondant réellement à l'abandon du programme. Toutefois, lorsque le bouton "Fini" se présente seul, la plupart des personnes handicapées comprennent qu'elles doivent "cliquer" sur celui-ci pour passer à la suite du programme (E31).

Par ailleurs, les termes utilisés dans les tableaux récapitulatifs des recettes et des dépenses ne sont pas très bien compris par tous les individus (E33 à E37) et le mot "solde" pose apparemment problème. Cependant, plusieurs éducateurs nous ont dit que ce récapitulatif intéresse beaucoup les personnes qui savent lire et qui le comprennent. Pour celles qui ne savent pas lire, il est impossible de le comprendre car il ne contient que du texte. Un paramètre existe d'ailleurs, permettant de le supprimer dans ce cas.

D'après les réponses aux questions (E5, E6, E19),les dessins sont en général très bien compris par l'ensemble de notre échantillon. Seule l'icône "oreille" est légèrement moins compréhensible (E11) et devrait sans doute être retravaillée.

En ce qui concerne le son, nous ne pouvons pas conclure grand-chose car la réponse à la question concernant la difficulté de compréhension des phrases prononcées avec un accent anglais (E51) ne semble pas refléter l'avis des utilisateurs (cfr 3.6 Critiques de l'outil questionnaire).

Nous avons ainsi une idée de la façon dont sont compris ces différents éléments de l'interface. Nous allons maintenant voir s'ils sont utilisés.

Quatre questions sont posées pour savoir si les personnes se basent sur ces éléments dans le programme. Dans notre échantillon, les dessins se sont révélés être les plus utilisés (E49:100 %), le texte (E48: 80 %) et les couleurs (E47: 75 %) le sont un peu moins et le son (E50: 60 %) se trouve en dernière position mais est de toute façon utile pour certains et doit donc être conservé. Si les phrases étaient prononcées "à la française", il serait sans doute plus employé. Notons que les éducateurs estiment qu'une icône représentant une oreille serait utile dans tous les écrans pour

la plupart des individus même si, paradoxalement, presque personne ne s'en sert.

Pour obtenir des renseignements supplémentaires à ce propos, nous avons réalisé une analyse en composantes principales en prenant 8 questions puisées dans les différentes rubriques du questionnaire psychologique (les mêmes que pour la première analyse en composantes principales) et 2 questions du questionnaire ergonomique qui sont les suivantes :

E47 : En général, la personne se base sur les couleurs pour se retrouver

E48 : En général, la personne se base sur le texte pour se retrouver

Cette seconde analyse est présente en annexe 8. Les deux premières composantes principales expliquent 70 % de l'information. Cela constitue un bon score. Le premier axe est lié positivement à l'ensemble des questions psychologiques et à la question E48. Il est lié négativement à la question E47. Il semble donc que le texte soit utilisé principalement par les personnes dont les capacités sont plus élevées alors que les couleurs aident plutôt les personnes qui ont plus de difficultés. C'est d'ailleurs normal puisque les acquis en lecture de ces personnes sont souvent faibles. Elles cherchent donc une aide autre que celle fournie par le texte pour s'orienter au cours d'une séance d'utilisation du logiciel. Ce résultat nous paraît intéressant et mérite d'être signalé.

Nous avons également tenté d'interpréter le deuxième axe mais sans grand succès. Cet axe est fortement lié positivement aux questions "peut distinguer les différentes rubriques d'un ticket de caisse" et "paie et vérifie la monnaie rendue sur 1000 FB" et négativement aux questions concernant les acquis en lecture et en écriture. On peut se demander s'il n'oppose pas des capacités assez "théoriques" (lecture/écriture) à des capacités plus "pratiques". Il opposerait peut-être les "débrouillards" aux "intellectuels". Ce n'est qu'une hypothèse qui devrait être confirmée par les psychologues ou les éducateurs. Nous ne connaissons en effet pas les individus et nous ne pouvons pas dire si cette distinction correspond à la réalité.

Certains éducateurs estiment que les postes de recette et de dépense sont trop peu nombreux et devraient dépendre des individus. Ainsi, des personnes qui vivent en institution et d'autres qui se débrouillent seules ne font pas les mêmes recettes et dépenses et auraient besoin de postes spécifiques.

Nous avons déjà réfléchi à ce problème avec des éducateurs et psychologues au cours de réunions de spécification de la gestion de budget. La solution consisterait à proposer toute une liste de postes parmi lesquels l'éducateur devrait choisir en fonction de l'individu grâce au programme de paramétrisation.

Parmi les 20 personnes handicapées mentales qui ont consulté l'état du porte-monnaie, 9 n'ont pas remarqué qu'il s'agissait de la somme qu'elles avaient enregistrée au début de la session. Il semble donc nécessaire d'essayer de remédier à cette situation puisque ce problème concerne presque la moitié de l'échantillon.

Tous les éducateurs estiment que l'écran d'état des comptes gêne beaucoup les personnes. En effet, cet écran apparaît très souvent et indique s'il y a équilibre des comptes ou pas. En cours de session, il est normal que l'équilibre ne soit pas directement atteint. Par conséquent, le message "Il manque... francs de recettes/dépenses" s'affiche à l'écran. Il donne l'impression aux personnes handicapées qu'elles ont fait une erreur et cet écran est donc perçu très négativement par celles-ci. Certains éducateurs nous ont dit qu'on perdait alors un des avantages de l'ordinateur qui est sa neutralité par rapport à un éducateur et qu'il faudrait présenter un tel écran uniquement en fin de session ou éventuellement dans la consultation de l'état du porte-monnaie de manière à ne pas "traumatiser" les utilisateurs.

Le logiciel contient un thermomètre qui a été conçu pour jouer plusieurs rôles:
- il doit permettre de visualiser l'état du porte-monnaie: une barre jaune indique ce niveau;
- il doit permettre de visualiser la somme des recettes moins la somme des dépenses: cette quantité est représentée par une surface colorée en bleu à l'intérieur du thermomètre;

- il doit permettre de comparer l'état du porte-monnaie à cette quantité de manière à vérifier s'il y a équilibre des comptes ou pas: ceci peut se faire en comparant le niveau de la barre jaune au niveau atteint par la surface bleue;
- il doit permettre de visualiser le dernier montant enregistré et donc constituer un feed-back de l'opération: la surface bleue contenue dans le thermomètre varie proportionnellement à la somme enregistrée, elle monte ou descend selon qu'il s'agisse d'une recette ou d'une dépense;
- il pourrait permettre d'apprendre des notions telles que "plus grand", "égal" : ceci peut se faire en comparant les niveaux et en observant l'ampleur des variations dans le thermomètre.

Nous avons posé plusieurs questions à propos de ce thermomètre car il nous semblait difficile à comprendre. Apparemment, nous ne faisions pas erreur car la plupart des utilisateurs n'ont pas le temps de le voir fluctuer (E39, E40, E53) et ne comprennent pas bien la signification des fluctuations (E55) ni des couleurs employées (E41). Seule la barre jaune qui indique le niveau atteint par l'état du porte-monnaie (E56) semble être un tout petit peu mieux comprise. Cependant, même cet élément pose problème puisque la majorité des individus ne saisissent pas sa signification. Enfin, personne n'a pu percevoir son erreur grâce au thermomètre (E30) alors qu'un des rôles de celui-ci est d'indiquer l'existence d'un déséquilibre (s'il y en a un). Il semble donc que l'utilisation du thermomètre doit être remise en cause...

Pour essayer de remédier à ce problème et d'y apporter une solution, nous avons beaucoup discuté du thermomètre avec plusieurs éducateurs afin de mettre en évidence les causes de son échec.

Il est apparu que le thermomètre fluctue trop vite. Cependant, il semble bien que, même s'il fluctuait plus lentement, il ne serait pas beaucoup mieux compris.

En effet, le thermomètre est de petite dimension et les personnes handicapées mentales n'y prêtent aucune attention.

De plus, les rôles du thermomètre sont très nombreux et il est donc nécessaire de présenter énormément de choses fort complexes dans un espace réduit.

Nous avons vu que deux notions sont représentées par le thermomètre : d'une part, la somme contenue dans le porte-monnaie de la personne, d'autre part, le solde de la session précédente et la différence

entre les recettes et les dépenses. Les montants correspondant à ces différentes valeurs ne sont pas affichés.

En outre, les fluctuations sont compliquées ! Lors de la saisie de la somme initiale, le thermomètre monte en bleu et la barre jaune se fixe à cette hauteur. Lorsqu'on enregistre la première dépense ou recette, on retrouve le thermomètre dans un état différent de celui dans lequel on l'a laissé. La barre jaune indique toujours le même niveau, mais le thermomètre s'est curieusement vidé. C'est perturbant et incompréhensible. Chaque fois qu'une recette est enregistrée, le niveau du thermomètre monte en vert et à la fin de la saisie, la surface verte se transforme en bleu. De même, lors de l'enregistrement de chaque dépense, une surface rouge proportionnelle à la dépense recouvre la surface bleue du thermomètre à partir du haut. Ensuite, le thermomètre se vide de cette surface. Il est aisé de constater que le jeu des couleurs et des niveaux est fort complexe.

La petite taille du thermomètre entraîne un manque de précision dans les fluctuations. Ainsi, dans certains cas, lorsque le montant enregistré est trop faible, le thermomètre n'assure plus aucun feed-back. S'il y a un déséquilibre faible , celui-ci ne sera pas nécessairement visible : l'objectif principal du thermomètre n'est malheureusement pas atteint, ce qui nous paraît grave. Si les montants correspondant aux deux niveaux étaient affichés, ce sérieux problème serait évité.

Nous avons également remarqué qu'il y avait parfois des "débordements" hors des limites du thermomètre. En fait, le programme de paramétrisation permet de fixer une somme qui est "le montant que la personne handicapée mentale n'aura jamais", c'est à dire que ce montant doit être supérieur à l'état du porte-monnaie et à la somme du solde et des recettes de la personne. La valeur fixée pour ce montant relève de la responsabilité de l'éducateur. Ce montant, la limite supérieure du thermomètre, est présent au dessus de celui-ci et trouble d'ailleurs un certain nombre d'utilisateurs. Il arrive cependant que l'éducateur n'ait pas bien fixé ce montant, et que le thermomètre déborde. Dans ce cas, il est rempli totalement, mais rien ne permet de dire s'il y a un faible ou fort débordement, ni même s'il y en a un ! Evidemment, on peut objecter que ce cas se présente lorsque l'éducateur a mal choisi le montant maximum. Nous dirons qu'il n'est pas aisé de fixer cette somme. Si l'éducateur fixe ce niveau très haut, pour éviter tout débordement, on perd également de la

précision car, dans la majorité des cas, tout se passera alors dans une petite partie du thermomètre.

Un débordement vers le bas peut arriver, même sans "erreur" de l'éducateur : si la personne commence, comme elle en a le droit, par enregistrer ses dépenses avant ses recettes, le thermomètre n'offrira plus aucun feed-back. Dans ces deux cas de débordement, le thermomètre ne présente aucune utilité.

Une balance "traditionnelle" (avec deux plateaux) doit également indiquer s'il y a équilibre des comptes ou pas. Son rôle se limite à cette fonction contrairement au thermomètre. D'après une question (E43) que nous avions posée, la symbolique de la balance est mieux comprise que celle du thermomètre. Cependant, 9 personnes sur 20 ne l'ont pas comprise. Il semble donc que cette idée de balance pourrait être gardée mais il faudrait essayer de l'améliorer. Même si ces balances traditionnelles ne sont plus utilisées, les individus semblent encore les connaître.

Néanmoins, nous tenons à signaler que la balance ne peut pas rester telle qu'elle est. En effet, celle-ci est en équilibre lorsque l'état du porte-monnaie correspond à la somme des recettes moins la somme des dépenses. Or, le plateau de gauche contient la lettre R pour recettes et le plateau de droite, la lettre D pour dépenses. La balance semble donc indiquer qu'il faut comparer la somme des recettes à la somme des dépenses et que celles-ci doivent être égales pour que les plateaux soient en équilibre. En fait, c'est tout à fait faux et cela peut induire les personnes en erreur. Cette méprise est très grave car les personnes risquent alors de ne rien comprendre à la logique du programme ou de croire qu'elles la comprennent alors qu'elles font fausse route!

Pour apporter des éléments de réponse à cet objectif de l'évaluation, nous disposons d'un outil supplémentaire: l'interface homme-machine.

Nous pouvons utiliser les recommandations en matière d'interface homme-machine dans l'évaluation du logiciel Comptes de manière à valider ou infirmer les choix de conception des différentes interfaces.

Le style d'interaction utilisé dans le logiciel Comptes est la manipulation directe. Il semble logique d'avoir pris ce choix. Ben Schneiderman cite comme avantages à ce style d'interaction : la facilité d'apprendre, la facilité de retenir, la plus grande satisfaction subjective des utilisateurs, l'encouragement à l'exploration du logiciel. La manipulation directe semble être conseillée pour notre population d'utilisateurs. Mais peut-être l'utilisation de la sélection par menu pourrait-elle être utilisée de manière plus intensive qu'elle ne l'est . Pour répondre de manière certaine à cette question, il conviendrait sans doute de construire un autre programme Comptes répondant aux mêmes objectifs et possédant les mêmes fonctionnalités mais dont le style d'interaction serait basé sur la sélection par menu. Il faudrait alors évaluer ce programme et le comparer à notre logiciel.

Les outils matériels d'interaction utilisés sont la souris et le clavier. Comme nous l'avons vu, les expérimentations avec les personnes handicapées mentales nous ont appris que certains utilisateurs avaient quelques difficultés à utiliser le clavier ne fusse que pour entrer leur nom ou un montant alors qu'il semble que l'utilisation de la souris ne pose aucun problème à l'ensemble de la population handicapée malgré les craintes des éducateurs.

Nous menons maintenant une critique détaillée et constructive de l'interface du logiciel dans le but de se donner les moyens d'en construire une meilleure version. S'il apparaît, comme nous l'avons montré, que les objectifs assignés au logiciel sont en grande partie atteints, la première version du logiciel n'est pas pour autant parfaite.

Nous utilisons les règles d'or de Schneiderman non pas pour montrer que le programme les respecte bien à certains moments mais pour montrer les endroits où elles ne sont plus respectées.

Ainsi, si nous pouvons dire que sur un certain nombre d'aspects, l'interface du programme est cohérente, il n'en reste pas moins vrai qu'un certain nombre d'incohérences peuvent être révélées. Et ces non-respects des critères ergonomiques sont, à notre avis, d'autant plus importants qu'il s'agit d'un logiciel pour des personnes handicapées ayant des déficiences mentales. La lutte pour la cohérence nous semble, par exemple, plus capitale pour des utilisateurs handicapés mentaux que pour d'autres individus.

Les 8 règles d'or de B.Schneiderman :

1. Viser l'uniformité, la cohérence

1° Uniformité des couleurs

Il ne suffit pas de montrer, pour prouver que cette règle est respectée, que le concept de dépense est toujours associé à la couleur rouge et celui de recette à la couleur verte. Si nous voulons réellement aider la personne handicapée mentale grâce aux différentes couleurs, il faut plutôt vérifier que la couleur rouge est toujours associée aux dépenses et la couleur verte aux recettes. Ce n'est qu'à cette condition qu'on pourra parler d'uniformité des couleurs.

Or, il apparaît que la couleur rouge est aussi la couleur de fond utilisée dans presque toutes les boîtes de dialogue et qu'elle représente les jours à venir lors de la saisie du jour d'une recette ou dépense dans le cycle de la semaine. La couleur verte représente les jours précédents dans ce même écran.

Cette double ou triple signification pour ces deux couleurs aura pour effet de semer le trouble et la confusion dans beaucoup d'esprits. Un éducateur auquel nous présentions le logiciel nous a avoué avoir compris que les recettes correspondaient aux jours passés et les dépenses aux jours futurs...

Par ailleurs, la couleur de fond des différents écrans est tantôt bleue tantôt noire.

2° Uniformité de positionnement et de taille

Plusieurs manquements à cette règle de cohérence peuvent aussi être relevés.

La taille des écrans varie de façon importante.

La localisation du thermomètre n'est pas la même dans tous les écrans. Lors de la saisie d'une somme d'argent par billets, il se trouve au milieu de l'écran. Lorsque cette saisie se fait par clavier, il est à l'extrême droite de l'écran et à mi-hauteur. Lors de la consultation du porte-monnaie, sa position est encore différente: il est en haut et à droite de l'écran.

De même, la localisation de l'icône recette (dépense) varie: lors de la saisie d'un montant par billets, elle est en haut et au centre de l'écran et lors de la saisie par clavier, elle se trouve en bas à droite.

La position des boîtes de dialogue n'est pas non plus identique d'un écran à un autre et la position des boutons au sein de ces boîtes de dialogue n'est pas toujours uniforme.

3° Uniformité de formes

Lors de la saisie de l'état du porte-monnaie, 3 commandes sont proposées : "Efface" , "Recommencer" et "Fini". La forme des icônes "Efface" et "Fini" est la même alors qu'il était possible de trouver d'autres formes que le rectangle : le losange, le cercle, le triangle,...

Lors de la saisie d'une recette (dépense), 3 commandes sont alors proposées : "Efface" , "Quitter" et "Fini". De nouveau, les icônes "Efface" et "Fini" ont évidemment la même forme mais, en plus, l'icône "Quitter" a la même forme, la même couleur et la même position que l'icône "recommencer" présente lors de la saisie de l'état du porte-monnaie alors qu'elle a une fonction totalement différente!

4° Uniformité des termes

De l'exemple précédent, il ressort que pour trois options présentes sur le même écran, on a choisi tantôt une forme impérative ("Efface"), tantôt infinitive ("Recommencer" ou "Quitter"), tantôt un participe passé ("Fini").

Il convient de remédier à ces incohérences mais on peut aussi songer à augmenter ou à renforcer la cohérence dans le programme. On pourra par exemple associer une couleur au porte-monnaie puisqu'il y en a déjà une pour les recettes et une autre pour les dépenses.

2. Permettre des raccourcis aux utilisateurs experts.

Cette règle semble respectée dans une très large mesure.

Nous ajoutons aux arguments de Marc Déplechin et Gianni Strappazzon que les paramètres permettent à une personne handicapée mentale à l'aise avec le programme de ne plus devoir préciser pour

Lors de la saisie du jour d'une recette (dépense), aucun feed-back négatif n'est offert à l'utilisateur qui essaye de sélectionner un jour postérieur au jour actuel.

Enfin, comme nous l'avons vu, toute la logique du programme repose sur l'équation: solde + somme des recettes - somme des dépenses = état du porte-monnaie. Il est donc indispensable, en début de session, d'indiquer la valeur du solde de la dernière séance afin que la personne puisse comprendre plus facilement la logique du programme. Or cette information (qui est un feed-back de la dernière séance) n'est présente que dans le deuxième écran de l'option "état du porte-monnaie" et est donc difficilement accessible.

4. Concevoir des dialogues permettant de découper la tâche de l'utilisateur

Cette règle est parfaitement bien respectée à notre sens et ne souffre d'aucune exception.

5. Une erreur ne doit jamais entraîner de résultats catastrophiques et l'utilisateur doit pouvoir corriger facilement

L'utilisateur peut toujours corriger une erreur qui vient d'être commise. Dans certains cas, ceci peut se faire en actionnant une commande telle que "Efface" ou "Recommencer". Dans d'autres cas, la possibilité de corriger est offerte par le logiciel sous la forme d'une boîte de dialogue: l'utilisateur ne doit pas prendre l'initiative.

Par contre, il n'est jamais possible de corriger une erreur qui s'est produite dans un écran précédent celui où l'utilisateur se trouve.

De plus, dans l'écran de saisie d'une somme par billets, il n'est pas possible d'utiliser la commande "Efface" plusieurs fois de suite. On peut soit tout recommencer, soit effacer seulement le dernier billet ou la dernière pièce. Certains utilisateurs auraient aimé effacer plusieurs fois consécutivement.

6. Permettre le retour en arrière.

L'utilisateur peut toujours recommencer ou abandonner le travail qu'il est en train d'effectuer mais les retours en arrière successifs sont impossibles.

7. Offrir à l'utilisateur le sentiment de contrôler l'application

Cette règle n'est mise en pratique que par l'unique menu du programme. A ce moment seulement, l'utilisateur pourra poser un choix sur les actions qu'il veut mener.

Néanmoins, on peut se demander s'il est toujours opportun dans le cas d'utilisateurs handicapés mentaux de vouloir respecter cette règle et s'il ne vaut pas mieux au contraire guider et soutenir au maximum cet utilisateur qui pose parfois difficilement des choix. D'après ce que nous avons pu observer lors de nos contacts avec les personnes handicapées, il est souvent préférable de les guider.

8. Réduire la charge de mémorisation

Le logiciel ne semble pas demander aux utilisateurs d'efforts de mémorisation étrangers à la nature des tâches à exécuter.

LES INFORMATIONS PSYCHOLOGIQUES DIVERSES

12) Quelle est la durée d'apprentissage du logiciel ?

L'expérimentation que nous avons menée est de trop courte durée pour que nous puissions répondre à cette question.

De plus, il faut bien être conscient que nous ne pourrons jamais déterminer une durée d'apprentissage valable pour toutes les personnes handicapées mentales. Elle change évidemment d'un individu à l'autre. Nous pourrions donc en donner une idée vague ou une moyenne.

Nous avons cependant tenté de voir si 4 séances permettraient un apprentissage significatif du logiciel par les personnes handicapées mentales. Notons bien qu'il ne s'agit pas de vérifier si le logiciel joue un

rôle en tant qu'outil d'apprentissage mais bien si les personnes handicapées mentales progressent vers une maîtrise du logiciel.

Nous avons considéré le questionnaire ergonomique dont nous avons éliminé un certain nombre de questions :

- les questions 8,15,16,17,27,28,32 et 44 auxquelles tous les individus n'ont pas répondu;
- les questions 22,23,25,26,47,48,49 et 50 qui ne semblent pas correspondre nécessairement à un apprentissage du logiciel.

Il reste donc 42 questions. Pour chacune de ces questions, nous appliquons la convention que nous avons expliquée au point 3.3.2.2 pour représenter les réponses :

- aux réponses positives correspond le chiffre 1;
- aux réponses négatives, le chiffre 0;
- lorsqu'il y a 3 possibilités, la réponse intermédiaire correspond à 1/2.

Pour certaines questions à connotation négative, la cotation se fait dans l'autre sens car une réponse positive correspond à un problème, à une difficulté.

Pour chaque individu, il est possible de calculer la somme des réponses aux questions considérées. On peut faire ceci pour la première séance d'utilisation et la quatrième par exemple. Un progrès d'un individu se traduira automatiquement par une croissance de son "score" entre la première et la quatrième séance. Nous avons donc des couples de données associés à des individus. Nous pouvons donc traiter ces "données pairées".

Nous faisons la différence entre le "score" obtenu pour la quatrième séance et le "score" qui correspond à la première séance d'utilisation pour ensuite estimer la moyenne des différences (d) et calculer un intervalle de confiance pour cette moyenne des différences.

Nous signalons qu'on assistera presque toujours à un progrès de la personne car, généralement, un recul d'une séance à l'autre est quasiment impossible. Si on observe une régression, cela peut cependant s'expliquer par le fait que les personnes handicapées mentales ont des comportements assez différents d'un jour à l'autre.

Nous pouvons également être plus en forme un jour qu'un autre mais chez les personnes handicapées mentales, les différences sont nettement plus accentuées. On pourrait avoir une séance durant laquelle rien ne va ! Une régression correspond donc à un "accident".

Nous présentons les scores que nous avons obtenus ainsi que la différence entre ceux-ci.

Les différences constituent les observations que nous supposons extraites d'une population normale.

individu	Σ des réponses lors de la 1ère séance	Σ des réponses lors de la 4ème séance	Différence (4-1)
A	11	18,5	7,5
B	23	32	9
C	26	32	6
D	13,5	16	2,5
E	24,5	30	5,5
F	32,5	32	-0,5
G	27,5	31,5	4
H	30,5	30,5	0
I	18,5	18,5	0
J	26	26	0
K	22,5	24,5	2
L	27,5	32,5	5
M	24	31	7
N	25	25	0
O	25	25	0
P	34,5	34,5	0
Q	15	33,5	18,5
R	25	38	13
S	21	37	16
T	32	32	0

Calculons maintenant

$\partial^{\wedge} = \Sigma d_i / n = 4{,}775$; l'estimateur de la moyenne des différences;

$s^2_d = \Sigma (d_i - \partial^{\wedge})^2 / n-1 = 32{,}38$; la variance de la moyenne des différences;

et $s_d = 5{,}69$; l'écart type.

En fixant le niveau d'incertitude à $\alpha=0{,}05$, on peut calculer un intervalle de confiance pour la moyenne des différences :

$$Pr\ [\partial^{\wedge} - Q_{tn-1}(1-\alpha/2)s_q/\sqrt{n} \leq \partial \leq \partial^{\wedge} + Q_{tn-1}(1-\alpha/2)s_d/\sqrt{n}] = 1-\alpha$$

$$\Pr [4{,}775 - Qt_{19}(0{,}975)5{,}69/\sqrt{20} \leq \partial \leq 4{,}775 + Qt_{19}(0{,}975)5{,}69/\sqrt{20}] = 0{,}95$$

$$\Pr [4{,}775 - 2{,}093 \,.\, 5{,}69/4{,}47 \leq \partial \leq 4{,}775 + 2{,}093 \,.\, 5{,}69/4{,}47] = 0{,}95$$

$$\Pr \; [2{,}112 \leq \partial \leq 7{,}438] = 0{,}95$$

Il apparaît ainsi qu'il y a un "progrès global" significatif puisque l'intervalle de confiance de la moyenne des différences ne recouvre pas zéro pour un niveau d'incertitude $\alpha = 0{,}05$.

Cependant, il n'est pas du tout évident que les observations soient extraites d'une population normale. Il serait donc préférable d'utiliser une méthode non-paramétrique. Nous pouvons estimer la médiane de la population. Nous suivrons, pour ce faire la théorie exposée dans la partie 3.3.2.5.4.

Pour cela, nous devons classer les d_i par ordre croissant et nous obtenons:

$d_1 = -0.5$	$d_{11} = 4$
$d_2 = 0$	$d_{12} = 5$
$d_3 = 0$	$d_{13} = 5.5$
$d_4 = 0$	$d_{14} = 6$
$d_5 = 0$	$d_{15} = 7$
$d_6 = 0$	$d_{16} = 7.5$
$d_7 = 0$	$d_{17} = 9$
$d_8 = 0$	$d_{18} = 13$
$d_9 = 2$	$d_{19} = 16$
$d_{10} = 2.5$	$d_{20} = 18.5$

Nous pouvons maintenant calculer :

$$\hat{m}ed = 1/2\,(d_9 + d_{10}) = 2.25$$

et chercher r tel que : $(1/2)^{20} \sum_{i=r}^{20-r} C^i_{20} \geq 0.95$

si on choisit un niveau d'incertitude $\alpha = 0.05$.

On obtient r=6 ce qui donne $(1/2)^{20} \sum_{i=6}^{14} C^i_{20} = 0.9586$.

L'intervalle de confiance pour la médiane est donc :

$$Pr\ [0 \leq med \leq 6] = 0.9586.$$

Nous pouvons donc dire, avec un niveau d'incertitude $\alpha = 0.05$, que la médiane de la population est comprise entre 0 et 6.

13) Quel type de personnes handicapées mentales peut utiliser le logiciel Comptes ?

Afin de répondre à cette question, il faudrait idéalement constituer un certain nombre de groupes de personnes handicapées mentales dont certains correspondraient aux prérequis et d'autres pas. On pourrait ainsi comparer les expérimentations faites avec ces différents groupes d'individus et déterminer si les prérequis ont été bien fixés ou s'ils doivent être revus. Ils peuvent être trop sévères ou pas assez.

Nous n'avons pas pu mener l'expérimentation de cette façon. Nous avons donc tenté de faire des groupes d'individus au sein de l'échantillon sur base des réponses au questionnaire psychologique.

Nous avons, par exemple, pris la rubrique "acquis en lecture" qui comprend 6 questions et compté pour chaque individu le nombre de réponses positives à ces questions. Sur base de ce nombre, nous avons séparé l'échantillon en deux groupes : le premier groupe est constitué des personnes pour lesquelles le nombre de réponses positives est inférieur ou égal à 2 et le deuxième groupe est constitué des autres individus. Le premier groupe compte 4 individus et le second 16 individus pour lesquels le nombre de réponses positives est supérieur ou égal à 4. Nous avons donc 4 personnes dont les acquis en lecture semblent assez faibles par rapport aux autres personnes. Nous pouvons essayer de déterminer si ces personnes rencontrent beaucoup de difficultés par rapport aux autres. Nous avons remarqué que 3 personnes sur 20 éprouvaient des difficultés pour reconnaître les dessins de pièces et de billets (E5,E6) et que ces 3 personnes faisaient justement partie du petit groupe de personnes qui semblent faibles en ce qui concerne les acquis en lecture. On pourrait donc se dire que les pièces et les billets devraient ressembler à ce qu'il sont dans la réalité pour que le logiciel puisse être utilisé par des personnes qui ne savent pas lire. Nous avons également remarqué que peu d'individus ne se basent pas du tout sur le texte et ne comprennent rien au tableau récapitulatif des recettes et dépenses mais que ces 4 personnes font de nouveau partie de ces individus.

Il semble donc que des personnes dont les acquis en lecture sont faibles peuvent utiliser le logiciel mais elles ne constituent pas la "population idéale".

Remarquons que nous ne pouvons rien tirer comme conclusions sur base d'un groupe de 4 personnes.

Nous avons essayé de constituer d'autres groupes sur base d'autres rubriques mais nous n'avons obtenu aucun résultat intéressant.

14) Les effets imprévus

Au cours de cette expérimentation, nous n'avons relevé aucun effet non prévu.

Certains effets étaient imprévus lors de la première expérimentation avec le logiciel mais nous en avons bien entendu tenu compte pour cette évaluation du logiciel. On ne peut plus parler d'effets imprévus.

Toutefois, nous n'aurions jamais cru que nous pourrions assister à un progrès même très léger dans les acquis en lecture durant l'expérimentation. Pourtant, nous avons appris qu'une personne a progressé un tout petit peu dans ce domaine (cfr. Objectif 7). Il s'agit d'un effet imprévu à court terme mais prévisible à plus long terme et, par conséquent, les objectifs fixés pour l'évaluation prévoyaient déjà ce genre de changements.

Nous n'avons en fait connu aucun effet imprévu qui ne puisse se rattacher à un des objectifs de l'évaluation.

3.6. CRITIQUE DES OUTILS D'EVALUATION UTILISES

Suite à l'expérimentation que nous avons menée, nous sommes en mesure de tirer certaines critiques sur les outils d'évaluation que nous avons utilisés. Ces conclusions sont positives comme négatives et peuvent être fort intéressantes pour l'évaluation d'un programme pour personnes handicapées mentales. Les évaluateurs de tout autre programme pourront profiter de notre expérience et tenir compte des difficultés que nous avons rencontrées. D'autre part, certaines lacunes et imprécisions sont apparues dans nos résultats et peuvent s'expliquer par ces difficultés.

3.6.1. L'INTERFACE HOMME-MACHINE

Nous avons utilisé l'outil "interface homme-machine" pour répondre à certains objectifs d'évaluation du logiciel. Cette discipline de l'informatique permet, en effet, de spécifier et d'évaluer un logiciel du point de vue de son ergonomie.

Mais nous ne disposons d'aucune recommandation concernant l'évaluation d'un logiciel pour personnes handicapées mentales. Il conviendrait donc de s'assurer que les recommandations peuvent être effectivemnent utilisables pour notre projet. Ainsi s'il apparaît que certaines "règles d'or" semblent tout à fait pertinentes et que leur application doit être impérativement respectée, il semble néanmoins que d'autres devraient être remises en question (cfr 3.5.) dans le cadre d'un logiciel pour personnes handicapées mentales.

3.6.2. LES QUESTIONNAIRES

Les questions posées semblent avoir été bien comprises par l'ensemble des éducateurs.

Cependant, beaucoup de questions demandaient une réponse fort tranchée (c'est-à-dire "oui" ou "non") et cela a posé des problèmes dans certains cas. Ainsi, certains éducateurs sont "sortis" des choix que nous leur imposions en répondant de manière nuancée ("plus ou moins", "à moitié",...) ou encore, en ajoutant des commentaires. Pour certaines questions, nous nous étions déjà rendus compte de ce problème et nous avions proposé 3 modalités de réponses plutôt qu'un choix binaire. Dans tous les cas, ces 3 possibilités ont été utilisées, preuve qu'il était nécessaire d'étendre le choix au moins à ces 3 alternatives.

On pourrait d'ailleurs envisager d'étendre encore le choix car ces 3 possibilités se sont parfois révélées insuffisantes. Ainsi, nous posons plusieurs questions destinées à savoir si la personne handicapée se base sur les dessins, les couleurs, le son ou le texte. Chacune de ces questions offre 3 modalités de réponse (uniquement, en partie, pas du tout) mais celles-ci ne nous permettent pas toujours de savoir si une personne se base plus sur un élément que sur un autre. En effet, il se peut qu'une personne se base en partie sur le texte et sur les dessins mais qu'elle se base plus sur le texte.

Nous passons alors à côté de cette information qui peut être importante. Dans ce cas, une pondération plus fine aurait permis à l'éducateur de mieux exprimer dans quelle mesure la personne se base sur chacun des éléments et, par la même occasion, de les classer. En tout cas, il semble intéressant, si pas indispensable, de proposer plus de choix de réponses pour chaque question.

Par ailleurs, le questionnaire comporte un nombre très élevé de questions. Nous avons en effet choisi, avant l'expérimentation, de poser beaucoup de questions pour obtenir un maximum de renseignements. Mais plus la taille du questionnaire est élevée, plus l'éducateur est découragé de le remplir sérieusement. On pourrait alors perdre de la précision et récolter moins de questionnaires que prévu. Face à ces deux objectifs contradictoires, le problème est de déterminer a priori la taille optimale du questionnaire. Après l'expérimentation, nous nous sommes rendus compte que certaines questions avaient des réponses très fortement corrélées et pourraient dès lors se réduire à une seule. Mais il aurait fallu disposer de cette information avant l'expérimentation. Si 2 questions paraissent fort liées a priori, il n'est pas du tout sûr qu'elles le soient dans la réalité. Ceci est d'autant plus vrai que la population qui utilise le programme est une population de personnes handicapées mentales et que "notre logique" peut différer de la leur. Nous devons donc nous méfier de nos idées préconçues. Nous avons d'ailleurs relevé un certain nombre d'exemples nous confirmant que la prudence doit rester de rigueur.

Ainsi, dans le questionnaire ergonomique, nous avons posé 2 questions destinées à savoir si la personne handicapée mentale se base sur les couleurs : l'une, pour l'ensemble du programme et l'autre pour l'écran de saisie du jour d'une recette ou d'une dépense. Nous aurions pu nous attendre à ce que ces 2 questions soient très fortement corrélées et donc qu'une seule question suffise mais ce n'est pas le cas. Nous avons donc bien fait de la poser à 2 reprises et nous aurions même peut-être dû la poser pour tous les écrans où la couleur intervient. La question "La personne sait écrire son nom" qui est posée dans les parties psychologiques et ergonomiques du questionnaire illustre également nos propos. Cette question est posée 2 fois car, même si pour un enfant le passage du papier au clavier se fait facilement, ce n'est pas nécessairement le cas pour les personnes handicapées mentales ! Certaines ont d'ailleurs besoin d'une aide

prolongée dans le temps pour écrire leur nom au clavier alors qu'elles n'ont pas de problème sur papier.

La situation inverse peut également se présenter. Nous aurions alors 2 questions fortement corrélées que nous n'aurions jamais cru corrélées a priori et pour lesquelles nous ne trouvons pas de lien logique et explicable. Les questions "La personne sait écrire son nom" et "La succession des écrans semble poser des problèmes à la personne" sont fortement corrélées (0.85) et nous sommes incapables d'en déterminer la cause!

Malgré tout, notre propre logique et la réalité observée se rejoignent souvent. Ainsi, les questions "La personne reconnaît les billets" et "La personne reconnaît les pièces" sont parfaitement corrélées. Dans ce cas, une seule question aurait suffi.

Comme on le voit, il est assez difficile avant l'expérimentation, de prendre la décision d'écarter certaines questions parce qu'elles semblent "logiquement" proches d'autres questions.

Nous avons vu (cfr 3.3.2.3.) que certaines questions nécessitent une simple observation de la part de l'éducateur alors que d'autres demandent pour y répondre de poser des questions à la personnes handicapée ou de faire des exercices avec elle. Pour ce deuxième type de questions, nous avons constaté que la plupart des éducateurs se contentent de donner leur appréciation sans chercher à ce que celle-ci ne repose nécessairement sur des bases solides. Les éducateurs sont généralement soumis à de nombreuses contraintes (les horaires, le nombre de personnes dont ils doivent s'occuper et qu'ils ne peuvent abandonner pour prendre une personne à part,...) et nous ne pouvons donc rien leur reprocher.

Cependant, leur subjectivité peut biaiser les réponses au questionnaire. En fait, chaque éducateur se construit légitimement sa propre opinion du logiciel mais celle-ci peut se refléter avec plus ou moins d'intensité dans les réponses aux questions qui demandent une appréciation.

Nous avons relevé un exemple très frappant qui illustre ce fait. Dans deux institutions, l'accent anglais est considéré comme "non gênant"; dans une institution, il est "gênant" au début mais ne l'est plus dès la deuxième ou troisième séance et dans les autres institutions, il est invariablement "gênant". Cet exemple ne présente aucune exception: à chaque institution correspond une seule réponse identique pour tous les

utilisateurs !!! Il faut donc être conscient de l'existence de la subjectivité et mettre en oeuvre des moyens pour réduire son importance.
Par conséquent, il semble nécessaire de mettre en relation plusieurs outils d'évaluation afin d'obtenir des résultats plus sûrs et plus précis.

Pour toute une série de questions, l'avis de l'éducateur n'est pas fiable car il risque d'interpréter (trop) de choses. Plutôt que de poser ces questions à l'éducateur, pourquoi ne pas les poser directement à la personne handicapée?

Nous avons d'ailleurs utilisé comme outil le questionnaire à l'utilisateur mais, d'une part, nous n'avons pas assez insisté sur son importance auprès des éducateurs et d'autre part, celui-ci était très sommaire et ne pouvait être compris par des personnes handicapées mentales. A l'avenir, il conviendrait donc de mener une réflexion sur cet outil. On veillera à poser des questions qui puissent être facilement comprises par une personne handicapée mentale. Bien évidemment, les personnes handicapées mentales ne peuvent lire toutes seules les questions qui leur sont posées. Il faudra donc un "médiateur" qui sera chargé de lire les questions et de noter les réponses fournies par la personne handicapée mais l'action de ce médiateur est strictement limitée à cela. Il ne pourra en aucun cas influencer, diriger ou suggérer des réponses d'une quelconque manière (ne fusse que par l'intonation de sa voix). Les questions devront être élaborées sur base des objectifs de l'évaluation.

Ainsi, pour le logiciel Comptes :

- afin d'évaluer la compréhension des termes utilisés dans le programme, on demandera par exemple à l'utilisateur :
  - "Qu'est-ce qu'une recette?"
  - "C'est quoi l'équilibre des comptes?"
  - "Que veut dire porte-monnaie?"

- pour mesurer l'apprentissage et les nouveaux acquis, on lui demandera :
  - "Qu'est-ce que le programme t'a appris?"
  - "Qu'est-ce que tu ne savais pas faire avant d'utiliser l'ordinateur et que tu sais faire maintenant?"

- pour évaluer l'interface du logiciel, on pourra lui poser des questions telles que :

  - "De quelle couleur sont les dépenses?"
  - "Tu aimes mieux utiliser la souris ou le clavier?"
  - "Pour entrer une somme d'argent, préfères-tu utiliser les billets ou les touches de l'ordinateur?"

- pour évaluer le plaisir qu'il éprouve à utiliser le logiciel, on lui demandera ce qu'il aime et ce qu'il n'aime pas dans les différents écrans et si "c'est gai de faire ses comptes avec l'ordinateur?".

Comme nous l'avons vu, l'ensemble des questions est bien compris. Par ailleurs, les éducateurs n'ont pas jugé qu'il manquait des questions. Nous sommes donc assurés que le questionnaire remplit bien sa fonction de mesure des objectifs de l'évaluation.

Néanmoins, il y avait peut-être des questions inutiles car redondantes. Comme nous l'avons déjà préconisé (cfr 3.3.2.2), il serait intéressant d'effectuer un test de l'outil questionnaire avant l'expérimentation à proprement parler sur un échantillon représentatif des éducateurs. Un questionnaire sur l'outil questionnaire pourrait donc leur être remis. Nous en avons rédigé un qui est présenté à l'annexe 3.

3.6.3. LES TRACES D'UTILISATION

Les traces peuvent nous apporter des renseignements importants sur les problèmes que pourraient rencontrer les personnes handicapées mentales lors de l'utilisation du programme.

En effet, nous avons vu (cfr 3.3.3.4) qu'il est possible de déterminer les fréquences d'un certain nombre de transitions jugées "critiques" qui correspondent à des erreurs commises par les utilisateurs. Cependant, les fréquences des transitions critiques pour un groupe d'individus ne peuvent s'obtenir qu'au terme d'un long travail. Il serait donc avantageux de disposer rapidement de ces fréquences sans devoir accomplir une tâche fastidieuse.

De plus, une transition critique peut correspondre à différents types d'erreurs et, dans certains cas, il pourrait bien être impossible de déterminer le type d'erreur rencontré. Si, par exemple, une personne désire effacer un billet lors de la saisie d'une recette, cela peut signifier :
- qu'elle manipule difficilement la souris et qu'elle a "cliqué" sur un autre billet que celui qu'elle voulait sélectionner (erreur de manipulation);
- qu'elle reconnaît difficilement les billets dessinés à l'écran;
- qu'elle ne se souvient plus bien des billets qu'elle a reçus.

Or les enseignements à tirer peuvent varier d'un type d'erreur à l'autre. Dans notre exemple, on pourrait arriver à différentes conclusions selon les cas :

- l'utilisation de la souris doit être remise en question;
- les billets sont mal dessinés;
- la personne n'est peut-être pas apte à tenir ses comptes.

Pour se faire une idée du type d'erreur, il est évidemment intéressant de pouvoir faire correspondre à chaque trace une colonne du questionnaire c'est-à-dire regrouper toutes les informations qui concernent une séance d'utilisation du logiciel. Toutefois, il n'est pas toujours possible de le faire car les traces sont simplement numérotées et ne sont pas datées. Théoriquement, un numéro devrait suffire mais nous avons remarqué que, dans la pratique, ce n'est pas le cas. En effet, il peut arriver qu'un éducateur utilise le programme sous l'identité d'une personne handicapée, qu'une personne débute une session dans de très mauvaises dispositions et que l'éducateur décide alors de mettre fin à la session et de recommencer une autre fois,... Dans ces circonstances, l'éducateur ne remplit évidemment pas le questionnaire et ne mentionne pas ce qui est arrivé alors qu'un fichier "Trace" s'enregistre. Il peut donc parfois y avoir un décalage entre les traces et le questionnaire et il est pratiquement impossible de repérer "l'événement perturbateur". Dater les traces permettrait d'éviter cet ennui.

Nous avons constaté que les éducateurs ont tendance à aider beaucoup plus les personnes qui rencontrent de grosses difficultés que les autres. Ainsi, si une personne a beaucoup de problèmes, l'éducateur lui demandera si elle est bien sûre de ce qu'elle fait et l'empêchera sans doute de faire certaines erreurs. Il est bien évident que cette information ne se trouvera

pas inscrite dans les traces car aucune action erronée n'aura été réellement entreprise. Par contre, une personne qui se débrouille bien sera moins suivie et l'éducateur lui laissera commettre des erreurs pour voir si elle constate qu'elle s'est trompée ou si elle est capable de rattraper par elle-même son erreur. Elle aura alors des traces moins "bonnes" qu'une personne moins apte à employer le logiciel!

On peut donc dire que les traces sont d'autant plus significatives que l'aide apportée par l'éducateur est faible. Il conviendrait en tout cas de connaître pour chaque utilisateur le niveau d'aide accordé par l'éducateur. Nous avions d'ailleurs demandé aux éducateurs de nous communiquer ce niveau mais la plupart ne l'ont pas fait.

Par ailleurs, il serait également intéressant de savoir combien de temps se déroule entre chaque action posée par l'utilisateur et donc d'introduire un chronométrage dans les traces.

3.6.4 LES CONTACTS AVEC LES UTILISATEURS

Des contacts avec les utilisateurs s'imposent puisque ce sont les premiers intéressés par le logiciel. Assister à une ou plusieurs séances d'utilisation du programme avec des personnes handicapées mentales nous semble très important pour se faire une idée fort précise de la façon dont se passe l'interaction avec l'ordinateur. Comme dans tout projet informatique, le contact avec l'utilisateur joue en effet un rôle essentiel dans l'évaluation du logiciel.

Cependant, dans certains cas, la personne handicapée mentale est troublée par notre présence et ne se comporte pas de la même façon qu'habituellement.

Des contacts en dehors de l'utilisation du programme peuvent également nous apporter beaucoup d'information. Il est intéressant que ce genre de contacts puisse préceder l'expérimentation afin que la personne soit moins troublée par notre présence.

3.6.5. LES ENTREVUES AVEC LES EDUCATEURS

Les discussions que nous avons eues avec les éducateurs nous ont apporté et appris beaucoup de choses.

En effet, ces personnes sont constamment en contact avec les personnes handicapées mentales et les connaissent donc assez bien. Elles sont en mesure de nous apprendre si le programme est utile ou pas pour les personnes handicapées mentales dont elles s'occupent et quelle utilité il peut avoir (outil d'autonomie et/ou d'apprentissage). Elles peuvent comparer le logiciel à d'autres méthodes utilisées pour la tenue de comptes par les personnes handicapées mentales. Outre les critiques qu'elles soulèvent sur le logiciel elles adoptent souvent une attitude constructive en nous aidant à spécifier des solutions aux problèmes décelés.

De plus, elles peuvent nous fournir quelques renseignements concernant les personnes handicapées mentales ayant utilisé le programme de façon à ce que nous puissions plus facilement interpréter les résultats obtenus avec les autres outils. En fait, ces discussions nous paraissent extrêmement intéressantes mais il ne faudrait pas pour autant négliger l'apport des autres outils.

CHAPITRE IV : PRESENTATION DE LA SECONDE VERSION DU LOGICIEL DE TENUE DE COMPTES

"Parler sans avoir écouté est un monologue".

Dondeyne

L'évaluation du logiciel Comptes nous a permis de mettre en évidence que le projet répondait assez bien à ses objectifs mais qu'un certain nombre d'améliorations et d'extensions pouvaient, et parfois devaient, être entreprises. Nous avons donc modifié le logiciel en partant du code source de celui-ci.

Puisque la deuxième version du programme Comptes repose sur la première, nous n'exposerons ici que les éléments qui diffèrent.

Trois types de changements ont été apportés à ce programme.
Il s'agit :
- de la création de nouvelles fonctionnalités;
- de la mise au point de nouvelles interfaces pour des fonctionnalités existantes;
- d'améliorations de l'interface.

4.1. LES NOUVELLES FONCTIONNALITES

Nous avons ajouté deux nouvelles fonctionnalités à la première version du logiciel.

4.1.1. LA TENUE DES COMPTES SUR UN CYCLE D'UN MOIS

Dans la première version du programme, il était possible de tenir ses comptes sur un cycle d'un jour ou d'une semaine. Il est désormais possible aussi de le faire sur un cycle d'un mois. Le choix entre ces trois cycles se fait grâce au programme de paramétrisation. Nous avons ajouté cette fonctionnalité car il semble que le cycle mensuel correspond mieux à la réalité de beaucoup de personnes handicapées mentales (cfr. 3.5.) que les autres cycles.

En outre, cette fonctionnalité pourra évidemment être réutilisée pour la gestion de budget où l'existence du cycle mensuel est indispensable.

Nous présentons ci-contre l'écran qui correspond à cette nouvelle fonctionnalité.

SPECIFICATION DE L'INTERFACE ABSTRAITE

Il faut :
- indiquer à l'utilisateur qu'il doit communiquer le moment où il a reçu ou dépensé son argent;
- proposer les différents jours possibles;
- saisir le jour choisi par l'utilisateur;
- demander la confirmation de ce jour;
- saisir la réponse à cette demande.

SPECIFICATION DE L'INTERFACE CONCRETE

Qu'il s'agisse de la saisie du jour d'une recette ou d'une dépense, un seul élément diffère : la question posée à l'utilisateur. Dans le cas d'une recette, la question "Quand avez-vous fait cette recette ?" est affichée à l'écran et peut également être prononcée. Pour une dépense, cette question se transforme simplement en "Quand avez-vous fait cette dépense ?"


4 11 18 25
5 12 19 26
6 13 20 27
7 14 21 28
1 8 15 22 29
2 9 16 23 30
3 10 17 24
2
3
4
5
6
7
1
C'est bien le Mercredi 4 Juillet ?
OUI
NON

La présentation des différents jours est la suivante : l'entièreté du mois en cours est affichée à droite sur l'écran et le mois précédent est également affiché complètement sur la moitié gauche de l'écran. Cela se présente donc comme certains calendriers sur lesquels sont affichés deux mois par feuille. Pour expliquer clairement la cause qui nous a poussés à choisir cette représentation, nous prenons un exemple. Supposons que nous soyons le 12 juillet, il faut donc pouvoir accéder au minimum à tous les jours compris entre le 12 juin et le 12 juillet. Mais cette représentation du 12 au 12 ne serait absolument pas habituelle et risque de ne pas être compréhensible. Il nous a semblé préférable de ne pas afficher des parties de mois, mais bien l'entièreté de chaque mois.

A chaque jour correspond une icône de forme rectangulaire. Le nom du mois est inscrit au dessus de ces icônes, et les jours de la semaine (lundi,...) sont inscrits à gauche et à droite des icônes.

Comme pour le cycle d'une semaine, nous avons respecté la convention selon laquelle le jour actuel correspond à une couleur particulière, les jours précédents à une deuxième et les suivants à une troisième. Les icônes sont donc de différentes couleurs.

Pour indiquer le jour de son achat ou de sa dépense, la personne handicapée mentale utilise la souris et "clique" sur l'icône correspondante. Une boîte de dialogue apparaît alors, demandant la confirmation du jour choisi. Si l'utilisateur infirme, l'écran reprend son aspect de départ et la personne peut recommencer sa saisie; si elle confirme, elle passe à l'écran suivant.

4.1.2. LA CORRECTION DE L'ETAT DU PORTE-MONNAIE

Comme nous l'avons déjà vu (cfr. 2.6. et 3.5.), une erreur commise lors de la saisie de l'état du porte-monnaie a des conséquences néfastes, non seulement pour la session qui nous occupe mais également pour les suivantes. Nous avons donc décidé d'offrir une possibilité de correction de l'état du porte-monnaie. Il s'agit d'une nouvelle option accessible à partir du menu. L'utilisateur peut donc corriger l'état du porte-monnaie pratiquement à tout moment au cours d'une session.

SPECIFICATION DE L'INTERFACE ABSTRAITE

Le logiciel :
- présente la somme contenue dans le porte-monnaie telle qu'elle a été enregistrée;
- saisit le nouveau montant;
- demande confirmation du nouveau montant enregistré;
- saisit la réponse.

SPECIFICATION DE L'INTERFACE CONCRETE

Si la saisie s'est faite par souris, l'écran se trouve dans l'état où la personne handicapée mentale l'a laissé, c'est-à-dire que la partie droite de l'écran qui contient les feed-backs reprend l'ensemble des billets et des pièces que la personne avait sélectionnés, ainsi que le total de la somme enregistrée. La partie gauche de l'écran contient les mêmes éléments que pour une saisie "habituelle" de l'état du porte-monnaie. Si l'utilisateur constate qu'il n'y a pas d'erreur, il choisit la commande "OK" et peut terminer l'opération sans rien changer. S'il a oublié des pièces ou des billets, il peut directement "cliquer" sur ceux-ci : ils viendront s'ajouter à ceux qu'il a entrés précédemment. Par contre, s'il veut recommencer la saisie complètement, il peut évidemment choisir la commande "Recommencer".

Si la saisie s'est faite par clavier, l'écran se présente également dans l'état où on l'a laissé, c'est-à-dire qu'il affiche le montant initial et la boîte de dialogue offrant le choix entre "OK" et "Effacer". La personne a ainsi la possibilité de quitter sans avoir effectué de changement ou d'effacer le montant et de recommencer la saisie.

4.2. LES NOUVELLES INTERFACES POUR DES FONCTIONNALITES EXISTANTES

4.2.1. L'écritoire

Nous avons donné ce nom à une nouvelle interface associée à la fonctionnalité de saisie du nom. Elle permet à l'utilisateur d'entrer son nom grâce à la souris.

En effet, nous avons constaté (cfr. 3.5) que les saisies par clavier posaient quelques problèmes. Le fait de devoir taper sur la touche "return" pour terminer la saisie et sur la touche d'effacement pour effacer des caractères perturbait un grand nombre d'individus handicapés.

De plus, le fait de passer de la souris au clavier ou du clavier à la souris peut être désagréable. A fortiori pour des personnes handicapées mentales, il semble indispensable d'éviter au maximum ces passages car ils peuvent, en outre, être sources de confusions. Ainsi, certaines personnes handicapées tentaient en vain de taper sur le clavier quand elles devaient utiliser la souris, et inversement.

Par ailleurs, avec un jeu adéquat de paramètres, il était possible d'utiliser le reste du logiciel avec la souris. En permettant aux individus d'utiliser également la souris pour communiquer leur nom, le logiciel s'ouvre à une population plus large (entre autres, aux handicapés physiques).

Il est évident cependant que certaines personnes communiquent plus rapidement leur nom à l'aide du clavier qu'avec l' "écritoire". C'est la raison pour laquelle nous avons ajouté un paramètre supplémentaire "saisie du nom", pour lequel les deux options possibles sont : "par clavier" et "par souris".

SPECIFICATION DE L'INTERFACE ABSTRAITE.

L'interface abstraite n'est pas modifiée par rapport à celle qui a été définie pour la saisie par clavier. Pour rappel, il s'agissait :
- d'indiquer à l'utilisateur qu'il doit communiquer son nom;
- de saisir sa réponse et d'assurer le feed-back;
- de demander confirmation du nom communiqué;
- de saisir la confirmation.

SPECIFICATION DE L'INTERFACE CONCRETE

La question "Quel est votre nom ?" est posée à l'individu oralement et sur écran.

Nous avons dessiné un clavier (selon l'ordre AZERTY) ne reprenant que les 26 lettres de l'alphabet, le "é", le "è", le trait d'union et la barre d'espacement. Chaque caractère est représenté par une icône sur laquelle la personne "clique" pour communiquer son nom. Nous n'avons évidemment pas repris la touche "return" qui posait problème. Elle a été remplacée par l'icône verte "OK", qui est d'ailleurs utilisée à divers endroits dans le programme pour inviter l'utilisateur à terminer l'opération en cours. Dans le même ordre d'idée, la touche d'effacement (<-) est remplacée par l' icône "Effacer" qui, elle aussi, est déjà employée dans d'autres écrans. Ces deux icônes se trouvent sous le dessin du clavier. Lorsque la personne handicapée "clique" sur une lettre, celle-ci apparaît sous l'interrogation "Quel est votre nom ?". Dès que la personne a composé son nom, elle "clique" sur "OK" et une demande de confirmation "Est-ce bien juste ?" apparaît dans une boîte de dialogue.

4.2.2. LA CALCULETTE

Cette nouvelle interface est associée à la saisie d'un montant d'argent, qu'il s'agisse d'une recette, d'une dépense ou de l'état du porte-monnaie. Les motifs justifiant cette nouvelle interface sont les mêmes que pour l'écritoire.

L'intérêt de la calculette nous semble être plus important pour la saisie du montant d'une dépense que pour la saisie d'une recette car nous avons constaté que beaucoup de personnes handicapées mentales recopient un ticket de caisse ou les prix indiqués sur les marchandises achetées : il leur était dès lors plus facile d'utiliser le clavier que la souris (billets) pour communiquer leurs dépenses. Cette dernière possibilité les obligeait en effet à décomposer la somme en divers billets qui la composent. Il ne leur est donc pas possible d'utiliser la souris pour recopier un montant d'argent.

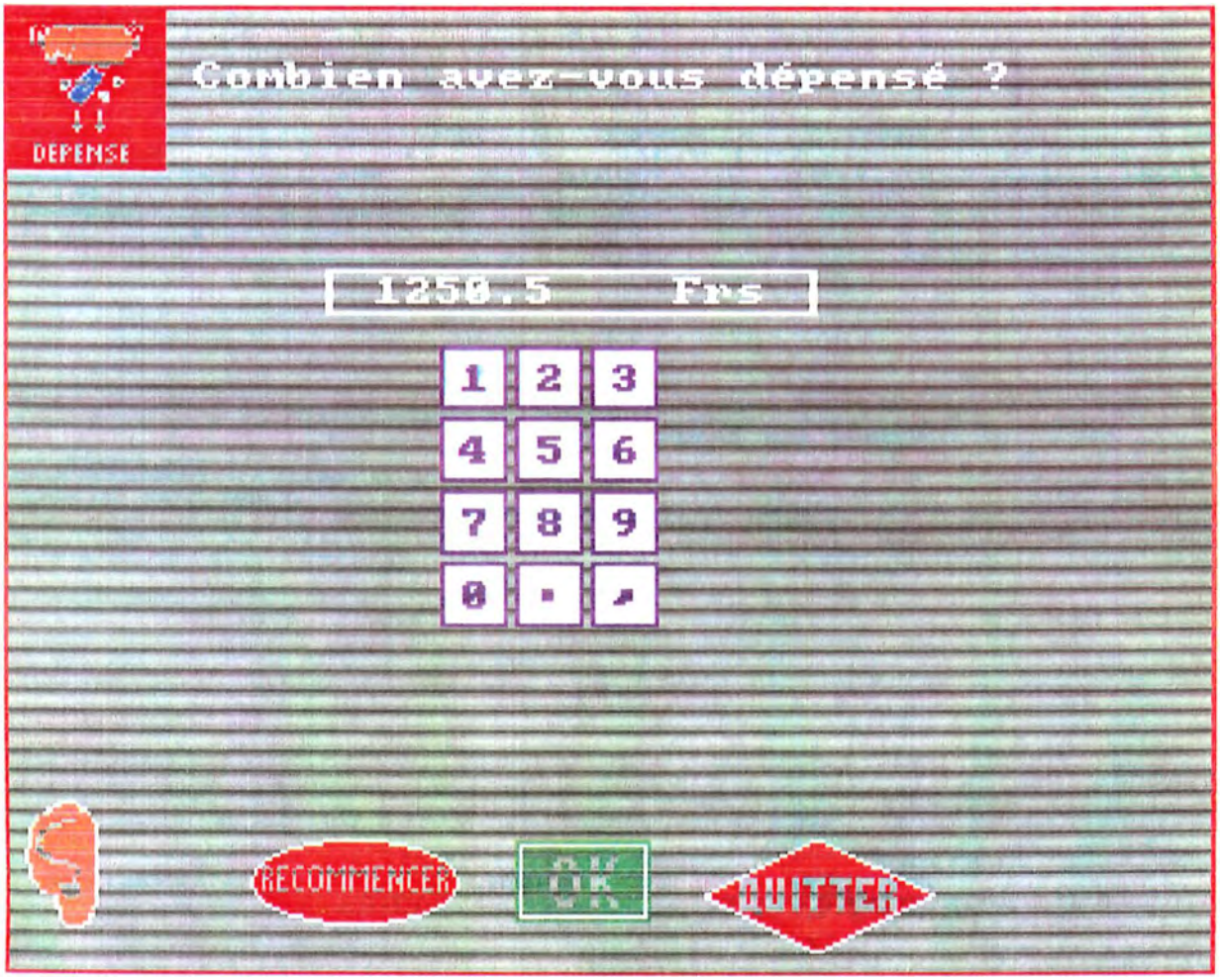
Nous présentons ci-contre l'écran qui correspond à cette nouvelle interface.

SPECIFICATION DE L'INTERFACE ABSTRAITE

L'interface abstraite ne change pas par rapport à celle qui a été définie pour la saisie d'une somme par clavier.
Il s'agit :
- d'indiquer à l'utilisateur qu'il doit communiquer la somme contenue dans son porte-monnaie ou le montant de sa recette ou de sa dépense;
- de saisir le montant et d'assurer le feed-back;
- de demander confirmation de la somme communiquée;
- de saisir la (non) confirmation de la somme.

SPECIFICATION DE L'INTERFACE CONCRETE

Selon que la saisie concerne l'état du porte-monnaie, une recette ou une dépense, une question différente est posée oralement et par écrit. Cette question est "Combien avez-vous d'argent ?", "Combien avez-vous reçu d'argent ?", ou "Combien avez-vous dépensé ?". Sous cette question, nous avons dessiné une partie de clavier correspondant aux touches numériques et comprenant le point décimal et la virgule. Chaque fois que la personne "clique" sur une des icônes numériques, le chiffre vient s'afficher au-dessus des touches, à la manière d'une calculette. Comme pour l'écritoire, la touche "return" et celle d'effacement ont été remplacées par deux icônes "OK" et "Effacer".


DEPENSE
Combien avez-vous dépensé ?
1250.5 Frs
1 2 3
4 5 6
7 8 9
0 .
RECOMMENCER
OK
QUITTER

Lorsque la personne a "cliqué" sur "OK", une demande de confirmation "Est-ce bien juste ?" apparaît dans une boîte de dialogue et la saisie de la réponse se fait en "cliquant" sur un des deux boutons "Oui" ou "Non".

Le programme de paramétrisation doit être employé pour décider de l'interface utilisée pour la saisie du montant du porte-monnaie, d'une recette ou d'une dépense. Désormais, les trois possibilités offertes sont celles-ci : billets, clavier et calculette.

4.2.3. LES REGLETTES

Les réglettes remplacent le thermomètre. Nous les avons classées dans la rubrique des nouvelles interfaces pour une même fonctionnalité. En effet, nous pensons que le thermomètre ne doit pas être uniquement considéré comme un élément de l'interface qui assure un feed-back lorsqu'une somme quelconque est enregistrée mais qu'il serait opportun de le classer dans les fonctionnalités du logiciel car il exerce une fonction bien particulière et importante, qui consiste à permettre à la personne handicapée de voir si son budget est en équilibre ou pas. Il ne s'agit donc pas d'un simple feed-back. Le thermomètre joue un rôle bien particulier ou, plus exactement, devrait jouer un rôle bien particulier, car, nous l'avons vu, le thermomètre ne fonctionne pas bien. Nous avons donc cherché une autre solution et, pour que celle-ci puisse donner pleine satisfaction, nous avons cherché un maximum de raisons et de critiques qui pourraient expliquer pourquoi le thermomètre ne remplit pas sa fonction. Nous avons exposé les rôles que devrait jouer le thermomètre ainsi que les causes présumées de son échec dans la section 3.5. Nous nous sommes basés sur ces éléments pour élaborer la nouvelle solution que nous présentons ici.

Tenant compte des difficultés liées au thermomètre, à sa petite taille et au manque d'attention des personnes, nous avons décidé de le supprimer et de le remplacer par les réglettes.

Celles-ci sont disposées sur un écran qui est appelé à plusieurs reprises au cours d'une session d'utilisation du logiciel, mais dont la présentation générale reste la même dans tous les cas : une "zone de travail"

occupe le quart supérieur de l'écran et le reste contient quatre réglettes. Chaque réglette correspond à un concept différent qui se caractérise par une couleur particulière.

La première réglette est verte et correspond à la somme du solde de la session précédente (considérée comme la première recette de la session en cours) et des recettes ultérieures. La deuxième réglette est rouge et correspond aux dépenses. La troisième est brune et représente la différence entre la première et la deuxième réglette c'est-à-dire le solde de la séance en cours. La quatrième réglette de couleur jaune correspond au contenu du porte-monnaie.

A côté de chaque réglette se trouve l'icône associée au concept, ainsi qu'un montant qui est la valeur représentée par la réglette. Une ligne servant d'échelle est affichée au-dessus de la première réglette. Sa valeur est fixée par l'éducateur grâce au programme "Paramètres" (c'est l'équivalent du niveau maximum du thermomètre).

La première réglette a une longueur proportionnelle au montant des recettes déjà enregistrées et commence à la même abscisse que la "ligne d'échelle". La deuxième réglette est proportionnelle aux dépenses déjà enregistrées. Elle ne commence pas à l'abscisse de l'échelle mais plutôt se termine où la réglette des recettes se termine. De cette manière, l'utilisateur est invité à opérer une soustraction entre la première et la deuxième réglette. Cette différence est matérialisée par la troisième réglette, qui commence donc à la même abscisse que la "ligne d'échelle" et a une longueur proportionnelle à la quantité :

somme des recettes - somme des dépenses.

La quatrième réglette représente l'état du porte-monnaie et commence à la même abscisse que l'échelle. Il y a donc équilibre des comptes lorsque la troisième et la quatrième réglettes ont même longueur.

Nous allons voir plus précisément à quels moments se font les appels à cet écran, et comment se matérialisent les fluctuations.

L'appel à ce nouvel écran se fait :

- après la saisie du nom

A ce moment, le solde de la session précédente est présenté. Nous avons décidé de mettre cet écran à cet endroit, car beaucoup de personnes ont regretté que le solde de la session précédente ne soit pas rappelé en début de session. Pour le connaître, il fallait absolument choisir l'option "Etat du Porte-Monnaie" dans le menu et consulter la première ligne du tableau récapitulatif des recettes. Ce solde constitue bien la première recette de la session.

La zone de travail contient la phrase : "La dernière fois, il restait ... Francs dans votre porte-monnaie" et, sous cette phrase, s'affiche une réglette verte correspondant au solde. Dans l'autre zone de l'écran, une réglette verte de la même longueur clignote et s'affiche en face de l'icône "Recette". De plus, le montant du solde s'inscrit à côté de cette première réglette.

- après la saisie de la somme contenue dans le porte-monnaie

Dans la zone de travail, la phrase : "Vous avez ... Francs dans votre porte-monnaie" s'inscrit et, en dessous, apparaît une réglette jaune représentant le montant du porte-monnaie. Dans la deuxième zone, cette barrette jaune se retrouve en face de l'icône du porte-monnaie, clignote à trois reprises, s'affiche et constitue la quatrième réglette. Le montant est également inscrit à côté de la réglette.

- après une éventuelle correction du porte-monnaie

Tout se passe exactement de la même façon qu'après la saisie du montant contenu dans le porte-monnaie.

- après l'enregistrement d'une recette

Dans la zone de travail, la phrase est : "Cette recette s'élève à ... Francs", et une réglette verte de longueur proportionnelle au montant de la recette s'inscrit sous cette phrase. Dans l'autre zone, cette recette vient

s'ajouter aux précédentes, la réglette correspondant à la nouvelle recette clignote à la droite des recettes précédentes et s'affiche, allongeant ainsi la première réglette. Le montant total des recettes augmente en conséquence. Si la personne a déjà effectué une ou plusieurs dépenses, la réglette "Dépenses" est déplacée vers la droite, de façon à venir toujours s'aligner par rapport à l'extrémité droite de la réglette des recettes. La troisième réglette, qui est la différence entre les recettes et les dépenses, s'allonge évidemment de la longueur de la nouvelle recette, et le montant qui y est associé est mis à jour.

- après l'enregistrement d'une dépense

La phrase de la zone de travail est cette fois : "Cette dépense s'élève à... Francs". Une réglette rouge se trouve sous cette phrase et représente la dépense en question.

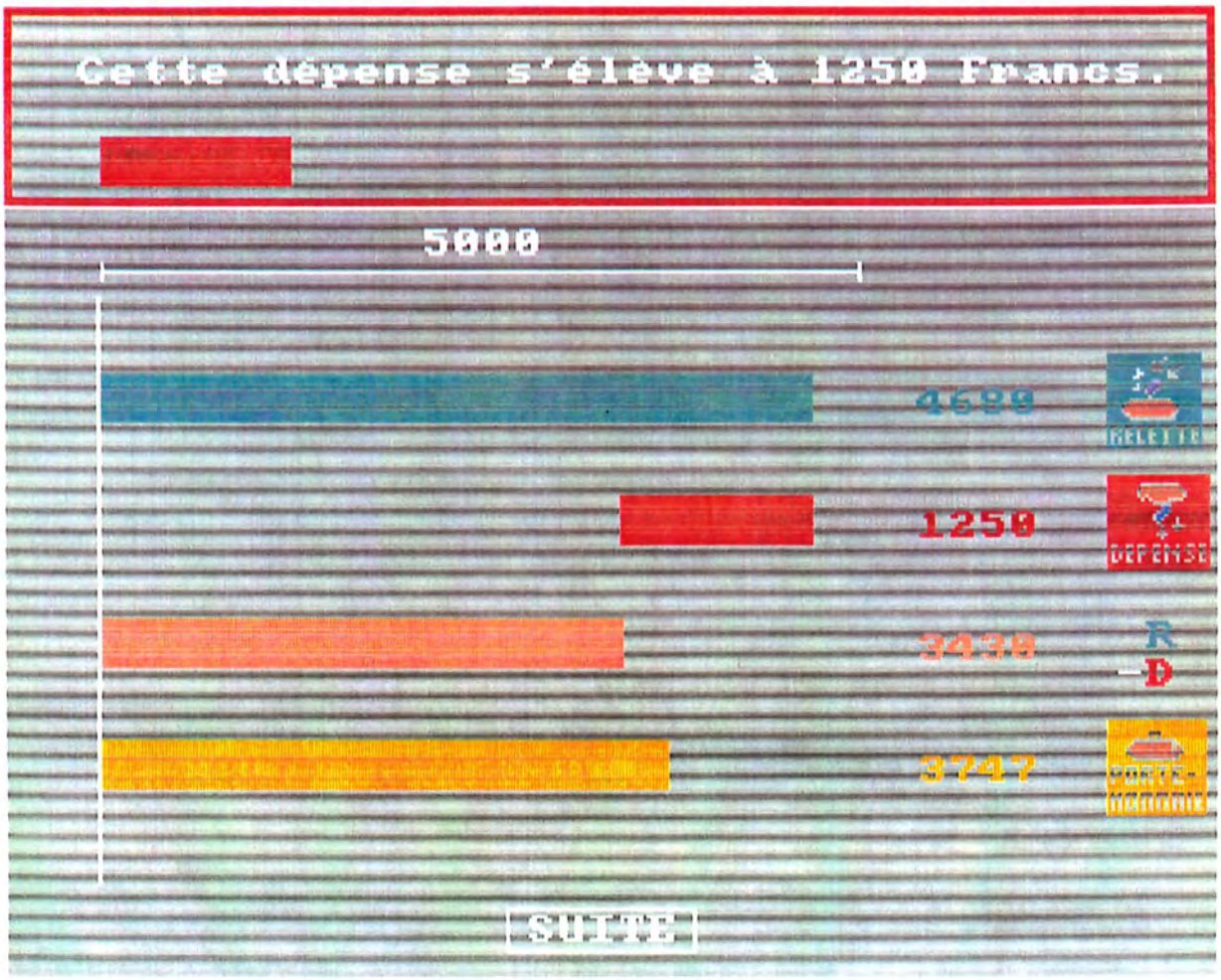
Dans la deuxième zone de l'écran, une barrette rouge de même longueur clignote, et s'ajoute à l'extrémité gauche de la réglette des dépenses. Pour donner une idée de soustraction des dépenses par rapport aux recettes, les dépenses vont dans l'autre sens que les recettes. Le montant des dépenses change en conséquence. La troisième réglette diminue du montant de la dépense effectuée.

- après avoir consulté l'état du porte-monnaie

Dans la zone de travail, la phrase "Vous avez ... Francs dans votre porte-monnaie" apparaît et la réglette jaune du porte-monnaie apparaît. Dans l'autre zone, cette réglette jaune clignote pour attirer l'attention de l'utilisateur.

- après avoir consulté le tableau récapitulatif des recettes

Dans la zone de travail s'affiche la phrase "Le total de vos recettes est de ... Francs" et l'entièreté de la réglette "Recettes". Dans l'autre zone, chaque recette clignote trois fois et s'affiche ainsi derrière la précédente, constituant peu à peu la totalité de la réglette "Recettes".


Cette dépense s'élève à 1250 Francs.
5000
4699
1250
3438
3747
SUITE

- après avoir consulté le tableau récapitulatif des dépenses

Tout se passe de la même manière pour les dépenses. La seule différence est le fait que les dépenses s'affichent une à une de droite à gauche, en commençant à la même abscisse que l'extrémité droite de la réglette "Recettes".

- après cet écran de consultation du tableau récapitulatif

L'écran d'état des comptes s'affiche avec pour but principal d'insister sur l'équilibre ou le déséquilibre. En effet, il nous semble indispensable d'insister sur cet élément. S'il y a équilibre, la zone de travail contient la phrase : "Bravo, il y a équilibre des comptes". Sinon, cette phrase devient : "Il y a un déséquilibre de ... Francs". Dans l'autre zone, qu'il y ait équilibre ou pas, les troisièmes et quatrièmes réglettes clignotent, de manière à inciter l'utilisateur à les comparer pour qu'il puisse prendre connaissance d'un éventuel déséquilibre, de son sens et de son ampleur.

- à la fin du programme, l'écran apparaît de la même façon que ci-dessus

Nous présentons un écran qui présente les réglettes ci-contre.

Après avoir présenté le fonctionnement des réglettes, nous sommes en mesure de citer quelques avantages de celles-ci par rapport au thermomètre.

Cet écran nous a permis de présenter facilement le solde de la session précédente.

Il attire beaucoup mieux l'attention qu'un thermomètre qui n'occupe qu'une petite partie de l'écran. L'instrument est beaucoup plus précis. Il fournit deux mesures supplémentaires : la somme des recettes, d'une part, et la somme des dépenses, d'autre part. Tout ne se passe plus dans une surface restreinte mais nous proposons quatre réglettes : l'information est plus riche puisqu'à chaque concept correspond une réglette. Par ailleurs,

l'information est plus précise puisque les montants sont affichés à côté des réglettes.

Grâce à l'indication des montants d'argent, même si le déséquilibre des comptes est minime, il apparaît.

Quand il y a "débordement" vers la droite ou vers la gauche, nous avons prévu de mettre des flèches qui indiquent "qu'il se passe quelque-chose" en dehors de ce qui est visible. De plus, l'utilisateur a une idée de la grandeur de l'élément qu'il vient d'ajouter en consultant la zone de travail.

Si une représentation sous forme de réglettes semble être préférable à l'emploi du thermomètre, il conviendra bien sûr d'évaluer sur le terrain dans quelle mesure ces réglettes apporteront une aide aux personnes handicapées mentales.

4.3. LES AMELIORATIONS DE L'INTERFACE

L'évaluation de l'interface de la première version du logiciel Comptes menée grâce aux questionnaires, aux entrevues avec les utilisateurs et les éducateurs et à une étude d'interface homme-machine, nous a permis de mettre en évidence un certain nombre de critiques. Nous avons tenté d'améliorer l'interface du logiciel sur base de celles-ci.

Afin d'améliorer la cohérence du logiciel, nous avons défini des règles précises concernant l'utilisation des différentes couleurs :

- la couleur de fond utilisée dans tous les écrans est le noir;
- la couleur rouge correspond toujours aux dépenses;
- la couleur verte correspond toujours aux recettes;
- la couleur bleue est associée aux interrogations : c'est la couleur de fond utilisée dans les boîtes de dialogue;
- les jours passés ne sont plus associés à la couleur verte mais à la couleur mauve et les jours à venir ne sont plus en rouge mais en gris.

Puisque nous avons remarqué que la présence de différentes couleurs dans le logiciel aide l'utilisateur, nous avons, le plus souvent possible, renforcé l'utilisation de ces couleurs de manière systématique et intensive.

Ainsi, nous avons associé au concept de "porte-monnaie" la couleur jaune ainsi qu'une icône de la même couleur.

De plus, tous les écrans qui se rapportent aux recettes (saisie du poste d'une recette, saisie du moment d'une recette, saisie du montant d'une recette, tableau récapitulatif des recettes) ont un cadre de couleur verte. Il en va évidemment de même pour les dépenses en rouge et l'état du porte-monnaie en jaune.

Cette utilisation intensive des couleurs est aussi présente dans l'écran des réglettes. Rappelons en effet que chacune de celles-ci est associée à une couleur particulière.

Nous avons veillé également à renforcer la cohérence des termes, des formes, des positions et des grandeurs. Ainsi, l'icône "Fini" a été remplacée par une icône "OK". Nous avons vu que les éducateurs estimaient que ce terme devrait être mieux compris par les utilisateurs. L'icône "Quitter" a désormais la forme d'un losange et se distingue donc maintenant de l'icône "Recommencer". La position des icônes "Recettes", "Dépenses" et "Porte-monnaie" est la même dans tous les écrans. La taille des différents écrans est uniformisée.

Nous avons veillé à augmenter l'implication de l'utilisateur. Ainsi, lors de la saisie ou de la correction d'un montant par billets, chaque pièce ou billet est dessiné dans une couleur très proche de celle qu'il a dans la réalité. La taille des différentes pièces et billets est proportionnelle à leur taille réelle, c'est pourquoi la pièce de 50 FB est plus petite que celle de 20 FB même si elle a une plus grande valeur monétaire. Enfin, le style d'écriture utilisé sur les dessins des billets est très proche de celui utilisé sur les vrais billets.

Nous avons amélioré la qualité des feed-backs offerts à l'utilisateur. Ainsi, dans l'écran de saisie d'une somme par billet, nous avions constaté que les "billets feed-back" s'affichant dans la partie droite de l'écran

posaient problème. Nous avons changé la manière dont se superposaient les billets de façon à offrir le feed-back le plus informatif possible.

Pour illustrer ces modifications, nous présentons aux pages suivantes l'écran de saisie du contenu du porte-monnaie (par billets) tel qu'il se présentait dans la première version du logiciel et tel qu'il se présente actuellement.

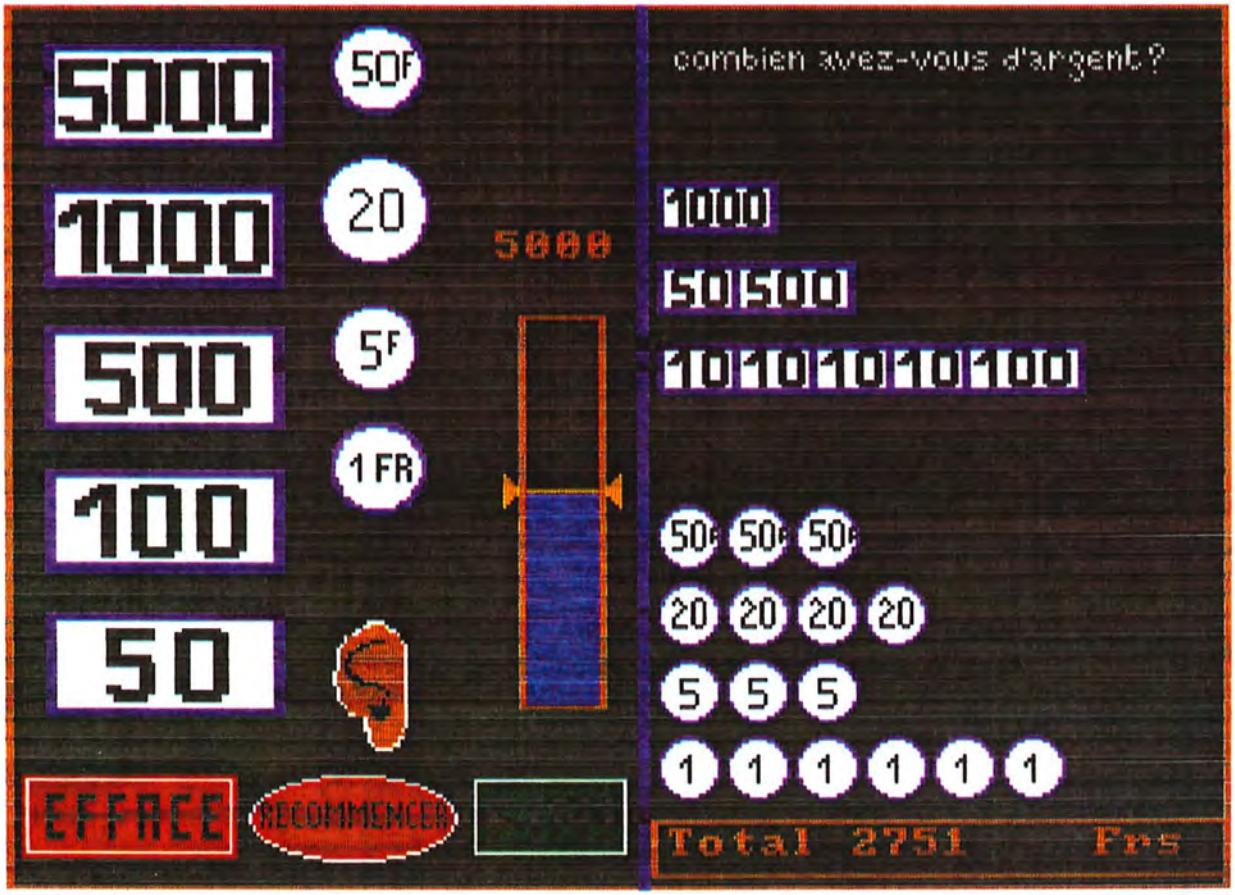
En fait, ces améliorations peuvent être d'une grande utilité pour la personne handicapée même si elles semblent peu importantes du point de vue de la programmation.

REMARQUES :

Tous les résultats de l'évaluation n'étaient pas encore en notre possession lorsque nous avons entamé la programmation de ces améliorations et extensions.

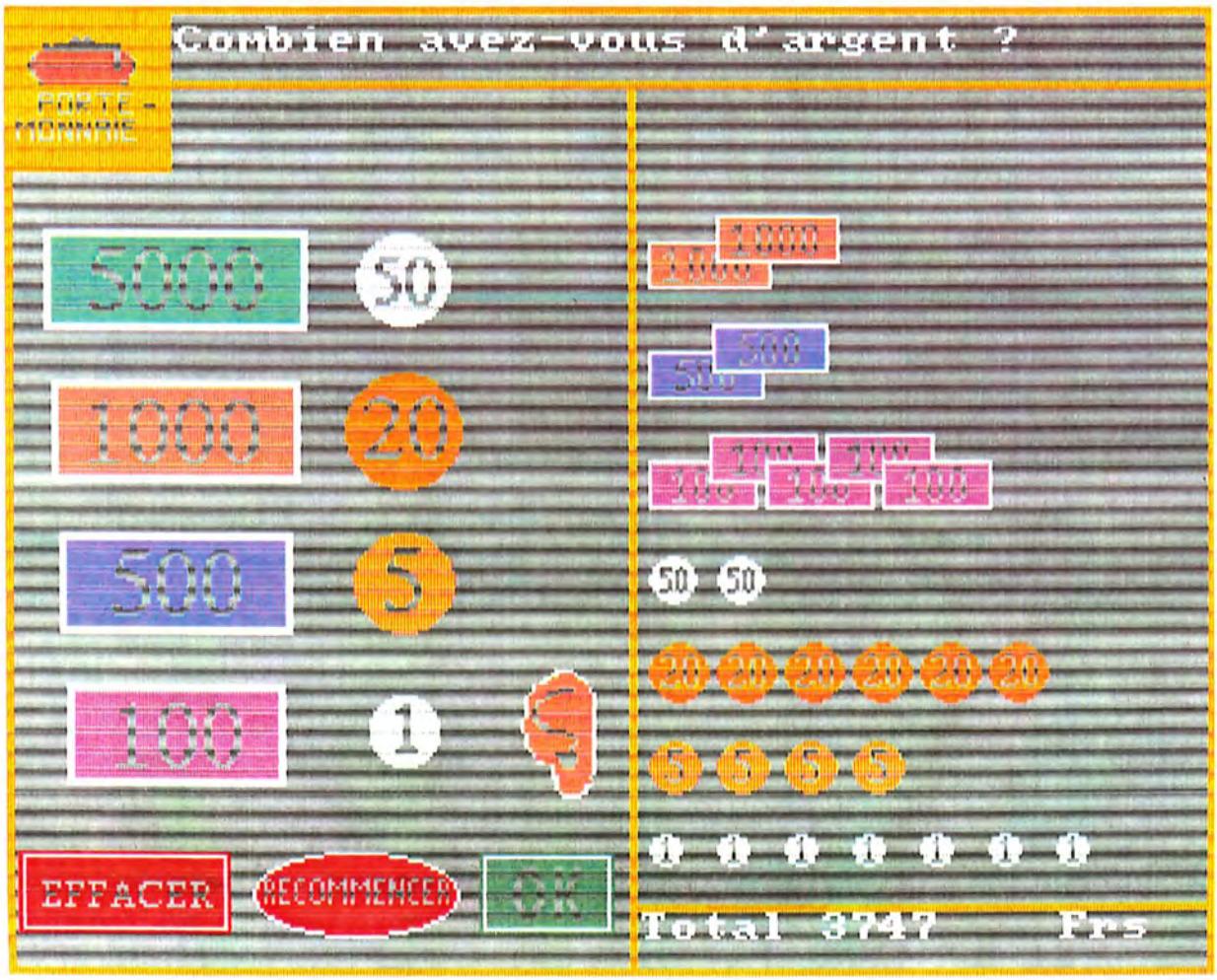
En outre, étant donné le temps limité qui nous était imparti, nous n'avons pas pu réaliser toutes les modifications et extensions souhaitées. Par exemple, pour améliorer la qualité sonore des phrases prononcées par le synthétiseur vocal de l'Amiga (destiné à prononcer des phonèmes de la langue anglaise), nous aurions pu digitaliser les phrases mais nous n'avons pas jugé cet objectif prioritaire. Les événements nous ont donné raison puisque la firme Commodore affirme que très prochainement le synthétiseur vocal de l'Amiga sera destiné à prononcer également des phonèmes français. Il suffira donc, à ce moment, de modifier la forme des intitulés des phrases et tout un travail de digitalisation et de programmation pourra ainsi été évité.

Nous ne prétendons donc pas avoir réussi à mettre au point un produit parfait. Des améliorations devraient encore être entreprises sur base des résultats de notre évaluation. Il conviendrait ensuite d'évaluer à nouveau le produit...



PREMIERE VERSION DU LOGICIEL COMPTES :

SAISIE DE L'ETAT DU PORTE-MONNAIE (PAR BILLETS)


Combien avez-vous d'argent ?
PORTE-MONNAIE
5000
50
1000
20
500
5
100
1
EFFACER
RECOMMENCER
OK
Total 3747 Frs

CHAPITRE V : AMELIORATION ET DEVELOPPEMENT D'OUTILS D'EVALUATION

Suite à l'évaluation du logiciel Comptes exposée dans le cadre du chapitre 3, nous avons pu apporter des extensions fonctionnelles et des améliorations ergonomiques au programme. Celles-ci ont fait l'objet du quatrième chapitre. Mais, comme nous l'avons vu, l'évaluation nous a permis aussi de critiquer les outils d'évaluation utilisés.

Le but de ce présent chapitre est de montrer au lecteur la façon dont nous avons pu améliorer et développer de nouveaux instruments d'appréciation à partir des critiques relevées.

En outre, ces améliorations et extensions concernant les instruments d'évaluation ont été intégrées à la seconde version du programme.

5.1. AMELIORATION DE L'EFFICACITE PRATIQUE DE L'OUTIL "TRACES"

Comme nous l'avons vu (dans la section 3.3.3.), les traces d'utilisation présentent un intérêt certain en tant qu'outil d'évaluation et peuvent être exploitées systématiquement en associant à chaque trace sa matrice des transitions effectives. Cependant, la construction d'une matrice correspondant à une session d'un utilisateur prend énormément de temps. Or, il est bien évident qu'une seule matrice ne peut nous fournir suffisamment d'information pour tirer des conclusions fondées à propos du logiciel. Il est nécessaire d'en construire beaucoup pour obtenir des résultats significatifs. Ce travail est donc extrêmement coûteux en temps et, en pratique, irréalisable.

Nous avons alors envisagé une première solution qui consistait à développer un programme qui, recevant en entrée un fichier "Trace", nous fournirait, en sortie, la matrice des transitions correspondante. Mais nous nous sommes heurtés à un gros problème: le fichier "Trace" tel qu'il a été conçu se prête fort mal à tout traitement systématique. Une analyse syntaxique des traces est, en effet, très difficile car elles n'ont manifestement pas été créées sur base d'un langage suffisamment structuré. Ceci est d'ailleurs tout à fait normal puisque l'objectif initial de ces traces

était de fournir à l'éducateur un compte-rendu fidèle des opérations effectuées par l'utilisateur sous la forme d'un texte.

De plus, l'ajout d'une fonctionnalité au logiciel Comptes pourrait modifier sensiblement un programme de construction de la matrice des transitions à partir des traces.

Par conséquent, nous avons préféré intégrer à la seconde version du logiciel Comptes l'enregistrement de la chaîne des états, c'est-à-dire la suite des états visités au cours d'une session. Un numéro identifiant chaque état possible, la chaîne des états se présente donc comme une suite de numéros. Après chaque séance d'utilisation, outre le fichier texte "Trace", un fichier contenant la chaîne des états est désormais enregistré.

L'information contenue dans une chaîne d'états est moins riche que celle d'une trace car cette chaîne ne comprend plus que les actions de l'utilisateur qui correspondent à un changement d'état. Toutefois, elle contient toujours l'ordre chronologique des transitions.

Il est facile de construire la matrice des transitions effectives à partir de cette suite d'états. Plus rien ne s'oppose ainsi à l'élaboration d'un programme qui construise la matrice des transitions. Celui-ci sera bien sûr indépendant du fichier "Trace" et exploitera le fichier contenant la chaîne des états.

De plus, si on ajoutait une nouvelle fonctionnalité au programme Comptes, il suffirait d'y faire correspondre un nouvel état. Cette méthode de construction de la matrice des transitions est générale puisqu'elle ne dépend ni des traces ni des fonctionnalités du logiciel Comptes.

Par ailleurs, nous avons daté les traces afin que la correspondance entre un fichier "Trace" et les informations contenues dans le questionnaire puisse désormais se faire aisément (cfr. 3.6.3).

5.2. DEVELOPPEMENT D'UN NOUVEL OUTIL : LA MESURE DES TEMPS.

5.2.1. PRESENTATION DE L'OUTIL

Les traces d'utilisation nous fournissent a posteriori une liste des actions menées lors d'une séance d'utilisation du logiciel. Très vite nous est apparue la nécessité de connaître le temps mis par un utilisateur pour accomplir telle ou telle action. C'est ce que font les psychologues lorsqu'ils

mesurent le temps séparant le stimulus (présentation à l'utilisateur d'un nouvel écran) et la réponse (action de l'utilisateur).

5.2.2. NOS MESURES DE TEMPS

Nous avons effectué deux types différents de mesures de temps.
Un premier type de mesures doit être mis en parallèle avec la chaîne des états. En effet, la seconde version du logiciel Comptes mesure et enregistre les temps passés dans chaque état visité par l'utilisateur au cours d'une session. Après chaque séance, cette information est inscrite dans un fichier (dont l'extension est .TPS). Chaque élément de celui-ci est un entier qui représente un temps exprimé en secondes. Supposons que les fichiers contenant la chaîne des états et les temps pour une séance soient les suivants:

contenu du fichier de la chaîne d'états	contenu du fichier des temps (en sec.)
1	10
2	8
5	**4**
8	7
5	3
11	7

Le nombre d'éléments des deux fichiers est toujours identique. Le $i^{ème}$ élément du fichier de la chaîne d'états correspond au numéro du $i^{ème}$ état visité et le temps de visite dans cet état est donné par le $i^{ème}$ élément du fichier des temps. Le contenu du fichier des temps est donc rangé dans l'ordre chronologique de visite des états. Ainsi, le $3^{ième}$ état visité lors de la session d'utilisation est l'état numéro 5 et la personne y est restée 4 secondes.

Pour évaluer le logiciel, la mesure des temps par état est bien sûr très intéressante, mais nous voulions également des mesures de temps qui aient une signification très claire pour les éducateurs. Il est sans doute plus utile pour ceux-ci de disposer d'une mesure des temps par écrans que par

états. Nous avons donc choisi 8 fonctionnalités pour lesquelles le logiciel Comptes (version 2) mesure et enregistre les temps. Nous y avons ajouté le temps total de la séance ainsi que le temps moyen nécessaire pour saisir un billet ou une pièce au cours de la saisie d'une somme par billets. En effet, un utilisateur peut enregistrer un seul billet comme 15 pièces et 12 billets. Le temps global pour cet écran de saisie peut donc énormément varier. C'est pourquoi nous avons fourni un temps relatif calculé par billet ou pièce entré.

Un fichier (dont l'extension est .TEMPS) contenant ces dix mesures de temps exprimées en secondes est enregistré sur disque après chaque séance. Ses éléments respectent l'ordre suivant :
1) temps nécessaire pour saisir le montant du porte-monnaie;
2) temps nécessaire pour saisir le montant d'une recette;
3) temps nécessaire pour saisir le montant d'une dépense;
4) temps nécessaire pour corriger l'état du porte-monnaie;
5) temps moyen nécessaire pour désigner une pièce ou un billet lors de la saisie du montant du porte-monnaie;
6) temps nécessaire pour saisir le moment d'une recette;
7) temps nécessaire pour saisir le moment d'une dépense;
8) temps nécessaire pour saisir le poste d'une recette;
9) temps nécessaire pour saisir le poste d'une dépense;
10) temps total de la séance.

5.2.3. UTILISATION DE L'OUTIL POUR REPONDRE AUX OBJECTIFS DE L'EVALUATION

Il est clair que la mesure des temps ne nous permettra pas, à elle seule, d'évaluer le logiciel. Elle devra être mise en rapport et liée aux autres outils. Cette remarque est d'ailleurs valable pour l'ensemble des outils.

De plus, les temps ne peuvent pas servir à classer une personne handicapée mentale par rapport à d'autres utilisateurs en fonction de ses "scores temporels". Les esprits lents ne sont sûrement pas moins "intelligents" que les plus rapides.

Cependant, nous considérons la mesure des temps comme un outil d'évaluation car elle nous fournit des indices supplémentaires pour répondre à certaines questions que nous pouvons nous poser à propos du logiciel Comptes. En outre, l'outil est caractérisé par une grande neutralité, comme les traces d'utilisation: il permet en effet d'éviter les biais induits par les éducateurs.

Nous espérons par exemple pouvoir évaluer la durée de la période d'apprentissage et/ou de familiarité avec le programme en comparant les temps mesurés sur plusieurs séances. A partir d'un certain nombre de séances, les temps devraient se stabiliser et signaler que l'apprentissage est terminé. Pourtant, avant de pouvoir conclure que le programme est compris, il faut encore s'assurer à l'aide d'autres outils que la personne n'agit pas par habitude. Elle pourrait, en effet, agir guidée par des automatismes sans pour autant avoir compris l'ensemble du programme.

Nous pourrions également déterminer quels sont les écrans que les utilisateurs comprennent le plus rapidement et le mieux en observant les temps consacrés à chaque écran par tous les utilisateurs. Dans ce cas, il faut veiller à comparer des écrans de même type. En effet, certains écrans nécessitent plus de temps que d'autres du fait des fonctionnalités qu'ils réalisent. Ainsi, on ne peut bien sûr pas comparer le temps passé dans un écran de confirmation (où on attend simplement de la personne qu'elle réponde par oui ou par non) à un écran où la personne doit entrer un montant d'argent. Par contre, il est tout à fait raisonnable de comparer l'écran de saisie du poste d'une recette à celui de saisie du poste d'une dépense. Cette comparaison pourrait nous livrer des renseignements précieux sur l'interface de ces deux écrans mais aussi sur les processus cognitifs mis en oeuvre par la personne handicapée mentale dans ces deux cas.

Un paragraphe du livre de Jean-Pierre Pourtois et Huguette Desmet intitulé "Epistémologie et instrumentation en sciences humaines" illustre bien, selon nous, les avantages et les inconvénients liés à la mesure du temps: "Quand, dans l'approche expérimentale classique, on introduit la mesure du temps, il s'agit d'un temps standardisé, d'un temps chronologique. C'est une mesure objective du temps qui s'écoule. C'est elle que l'on retrouve fréquemment dans les tests standardisés (évaluation du

Q.I., des aptitudes, des connaissances). La neutralité de l'observateur est sauvegardée. Le vécu du sujet, son histoire sont évacués."

5.2.4. METHODES D'EXPLOITATION

Il est possible d'appliquer des méthodes de traitement de données aux mesures de temps.

Nous pourrions par exemple y appliquer l'analyse en composantes principales de façon à résumer l'information qu'on peut tirer des temps et à déterminer en outre s'il existe une corrélation entre certaines mesures de temps.

L'outil d'appréciation "Mesure des temps" est donc un outil qui requiert la mise en place d'une expérimentation mais il ne nécessite pas l'intervention directe d'un accompagnateur.

En outre, il existe des méthodes formelles permettant de tirer des résultats des expérimentations faites avec cet outil.

Nous pouvons donc le situer dans la grille de positionnement des outils définie au début de la section 3.3 de la manière suivante :

	avec l'intervention de l'accompagnateur	sans l'intervention de l'accompagnateur
"formel"		Mesure des temps
"non-formel"		

5.3. CONCEPTION ET DEVELOPPEMENT D'UN PROGRAMME D'AIDE A L'EVALUATION

5.3.1. LES OBJECTIFS DU PROGRAMME

Le premier objectif de ce programme est d'offrir à l'éducateur un outil agréable et facile à utiliser lui permettant de se faire une bonne idée a

posteriori d'une séance d'utilisation de l'ordinateur. Le programme de dépouillement des traces doit ainsi lui permettre de laisser de plus en plus la personne handicapée seule avec l'ordinateur pour qu'elle puisse acquérir peu à peu sa véritable autonomie.

Le second objectif consiste à fournir à l'évaluateur un moyen simple et rapide pour le dépouillement des informations contenues dans les fichiers "Traces" et "Temps".

5.3.2. LES FONCTIONNALITES DU PROGRAMME

Après avoir entré un mot de passe, l'éducateur ou l'évaluateur communique le nom de la personne handicapée à propos de laquelle il désire obtenir de l'information. Un menu propose alors les cinq fonctionnalités suivantes :

1. le tableau des temps
2. le graphique des temps
3. les traces d'utilisation
4. la matrice des transitions
5. les transitions critiques

Il est également possible d'imprimer ces différents types de renseignements. Seul le graphique des temps ne peut l'être car chaque temps correspond à une couleur différente et il serait donc impossible de les distinguer sur une imprimante habituelle. Nous présentons ci-après quelques listings obtenus avec le programme "Traces" qui illustrent les différentes fonctionnalités.

1) Le tableau des temps

Ce tableau reprend pour un utilisateur donné les 10 mesures de temps définies au point 5.2.2. pour toutes ses séances d'utilisation du programme Comptes. Il fournit en outre pour chacune de ces 10 mesures, le minimum, le maximum et la moyenne pour les différentes séances. Il peut donc intéresser tout autant les éducateurs que les évaluateurs du logiciel.

2) Le graphique des temps

Nous avons décidé de présenter ces 10 mesures de temps sous la forme graphique. En effet, une représentation graphique est souvent fort parlante: les grandes tendances apparaissent au premier coup d'oeil.

Un premier graphique présente l'évolution du temps total des différentes séances d'un utilisateur. L'axe des abscisses correspond aux différentes séances et l'axe des ordonnées au temps total d'utilisation du programme par séance en minutes.

Un second graphique présente l'évolution des 9 autres mesures de temps. Le numéro des différentes séances s'inscrit sur l'axe des abscisses et les différents temps d'utilisation sur celui des ordonnées.

L'échelle utilisée pour les ordonnées n'est pas constante et est calculée pour chaque utilisateur de manière à présenter les temps avec la plus grande précision possible. C'est précisément en raison de problèmes d'échelle que nous avons séparé le temps total des autres : leurs ordres de grandeur sont très différents.

3) Les traces d'utilisation

Cette fonctionnalité présente à l'écran et/ou imprime sur papier le fichier "Trace" d'une séance d'utilisation en offrant la possibilité de "scrolling" du texte. Il était déjà possible d'obtenir ce fichier mais il fallait que l'éducateur entre dans l'interpréteur de commandes de l'Amiga (le CLI), se place sur le volume contenant les traces, obtiennent le répertoire de ces fichiers et enfin, lance une commande pour savoir le lire et une autre pour l'imprimer. De plus, il n'y avait pas moyen d'effectuer de "scrolling". Ceci nécessitait donc un minimum de connaissances sur l'Amiga et beaucoup d'éducateurs n'osaient pas se lancer dans cette "périlleuse aventure". Cette fonctionnalité illustre bien le fait que des développements assez simples du point de vue de la programmation peuvent être très intéressants et utiles.

4) La matrice des transitions

Il s'agit de présenter la matrice des transitions effectives pour une session d'utilisation comme nous l'avons définie au point 3.3.3.

Le programme commence donc par saisir le numéro de la session qui fera l'objet d'une analyse. Ensuite, pour des raisons pratiques de place disponible sur l'écran, il présente cette matrice en 4 parties. Il n'est en effet pas possible d'afficher en une seule fois les 32 lignes et les 32 colonnes correspondant aux 32 états du programme; l'impression de cette matrice sur papier est donc très utile car elle se présente sous un aspect beaucoup plus lisible. Cette fonctionnalité n'intéressera bien sûr que l'évaluateur.

5) Les transitions critiques

Le programme demande le numéro de la séance d'utilisation pour laquelle on veut consulter les transitions critiques. Il présente ensuite pour celle-ci la liste des transitions critiques et, pour chacune d'elles, les fréquences de passage par ces transitions. Cette fréquence pour la transition de l'état i à l'état j est donc le rapport entre le nombre de passages effectifs de l'état i à l'état j et le nombre de fois où cette transition à été susceptible d'être franchie (c'est-à-dire le nombre de fois où on est arrivé à l'état i).

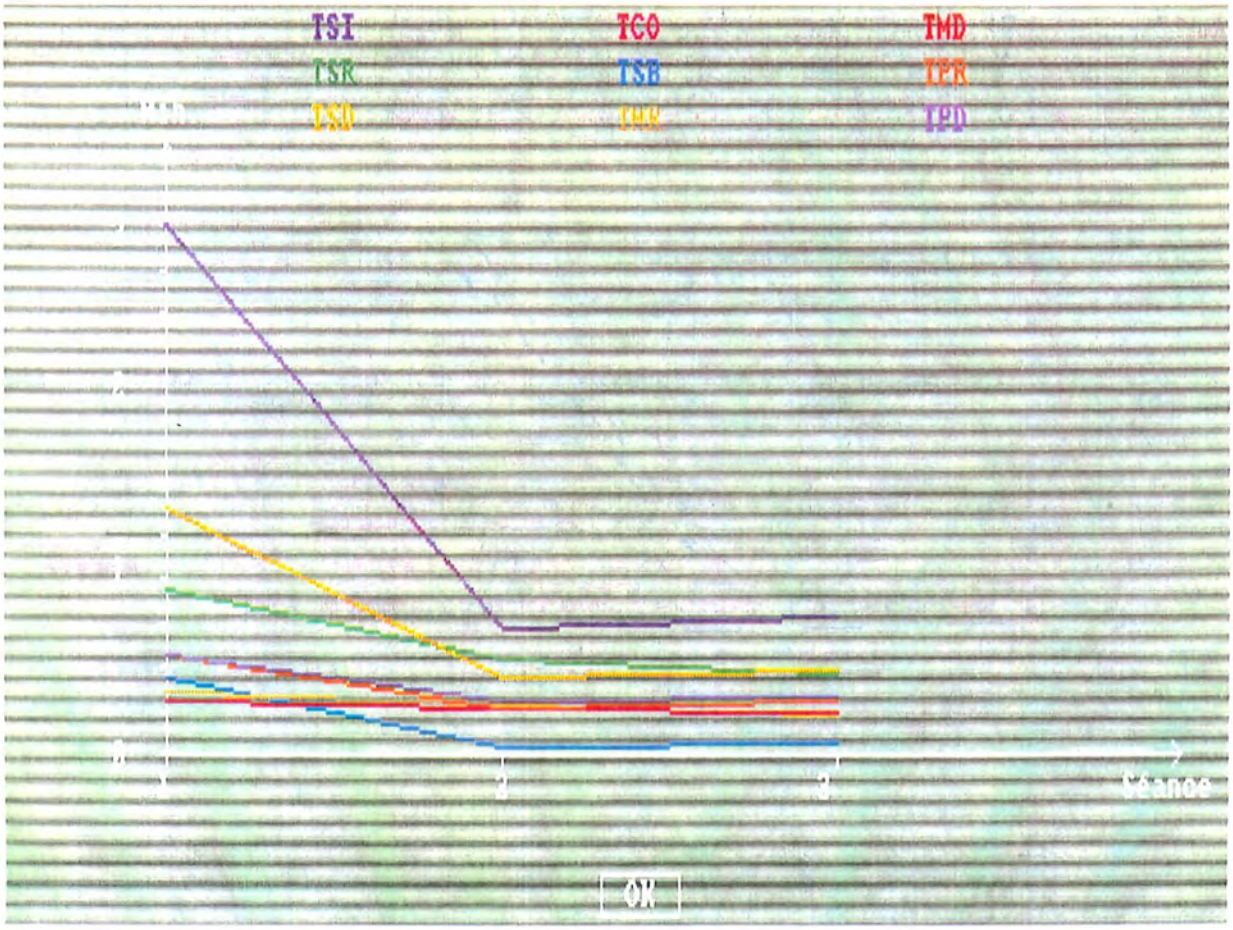
Nom de l'utilisateur : JANE

Date de l'impression : 10 Aout 90

Temps d'exécution

- TSI = Temps nécessaire pour saisir la somme initiale
- TSR = Temps nécessaire pour saisir le montant d'une recette
- TSD = Temps nécessaire pour saisir le montant d'une dépense
- TCO = Temps nécessaire pour corriger le montant initial
- TSB = Temps nécessaire moyen pour cliquer sur un billet lors de la saisie de la somme initiale
- TMR = Temps nécessaire pour saisir le moment d'une recette
- TMD = Temps nécessaire pour saisir le moment d'une dépense
- TPR = Temps nécessaire pour saisir le poste d'une recette
- TPD = Temps nécessaire pour saisir le poste d'une dépense
- TT = Temps total de la séance

	TSI	TSR	TSD	TCO	TSB	TMR	TMD	TPR	TPD	TT
Séance 1	180	56	84	---	27	21	19	34	34	651
Séance 2	43	33	26	---	3	17	16	16	19	254
Séance 3	47	28	29	---	4	14	15	19	20	240
Moyenne	90.0	39.0	46.3	---	11.3	17.3	16.7	23.0	24.3	381.7
Maximum	180	56	84	---	27	21	19	34	34	651
Minimum	43	28	26	---	3	14	15	16	19	240


TSI
TCO
TMD
TSR
TSB
TPR
TSD
TMR
TPD
Frames
OK

Nom de l'utilisateur : JANE

Numéro de la séance : 2

Date de l'impression : 10 Aout 90

TRACES D'UTILISATION

Date de la session : 3 Juillet 90
Il donne le nom : JANE
Il confirme le nom

Liste des paramètres choisis
Pas d'impression
Questions posées par écrit + parole
Saisie du Nom par clavier
Recettes saisies par billets
Dépenses saisies par calculette
Porte-Monnaie -> nbres+dessins+parole
Réglettes présentes
Notion du temps présente
Poste de dépense à spécifier
Récapitulatif présent
Cycle d'une semaine
Début de cycle = vendredi

Il appuie sur 100 Frs
Il appuie sur 100 Frs
Il appuie sur 100 Frs
Il appuie sur 100 Frs
Il appuie sur 100 Frs
Il appuie sur 100 Frs
Il appuie sur OK
Il confirme la demande
Il donne une s. initiale de : 600.00 fr
Il choisit la fonction : recettes
Il appuie sur travail
Il confirme
Il donne le poste : Travail
Il appuie sur vendredi
Il confirme
Il donne le jour : Vendredi
Il appuie sur 500 Frs
Il appuie sur 100 Frs
Il appuie sur OK
Il confirme la demande
Il donne une recette de : 600.00 frs
Il choisit la fonction : depenses
Il appuie sur alimentation
Il confirme
Il donne le poste : Alimentation
Il appuie sur samedi
Il confirme
Il donne le jour : Samedi
Il appuie sur 1
Il appuie sur 3
Il appuie sur 0

Nom de l'utilisateur : JANE

Numéro de la séance : 1

Date de l'impression : 10 Aout 90

MATRICE DES TRANSITIONS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Nom de l'utilisateur : JANE

Numéro de la séance : 4

Date de l'impression : 10 Aout 90

TRANSITIONS CRITIQUES

Lors de la saisie du nom :

- l'utilisateur désire recommencer la saisie du nom : 0 /2
- il infirme lorsqu'on lui demande si son nom est juste : 1 /2

Lors de la saisie de la somme initiale :

- il demande d'effacer la dernière opération : 2 /4
- il demande de recommencer la saisie : 1 /4
- il ne confirme pas la somme aprés avoir cliqué sur ok : 0 /1

Lors de la saisie d'une recette :

- il ne confirme pas le poste budgétaire qu'il a choisi : 3 /5
- il ne confirme pas le moment qu'il a choisi : 0 /2
- il demande d'effacer la dernière opération : 1 /4
- il désire quitter la saisie : 1 /4
- il ne quitte pas aprés l'avoir demandé : 0 /1
- il quitte réellement la saisie : 1 /1
- il ne confirme pas le montant après avoir cliqué sur ok : 1 /2

Lors de la saisie d'une dépense :

- il ne confirme pas le poste budgétaire qu'il a choisi : 1 /2
- il ne confirme pas le moment qu'il a choisi : 2 /3
- il demande d'effacer la dernière opération : 1 /2
- il désire quitter la saisie : 0 /2
- il ne quitte pas aprés l'avoir demandé : 0
- il quitte réellement la saisie : 0
- il ne confirme pas le montant après avoir cliqué sur ok : 0 /1

Lors de la correction :

- il demande de corriger : 0 /6
- il demande d'effacer la dernière opération : 0
- il désire recommencer la saisie : 0
- il ne confirme pas le montant aprés avoir cliqué sur ok : 0

Lors de la fin du programme :

- il ne confirme pas sa demande de fin de session : 1 /2
- il revient au menu après présentation de la balance : 0 /1

CHAPITRE VI : EVALUATION DE LA SECONDE VERSION DU LOGICIEL DE TENUE DE COMPTES

Nous avons présenté dans le chapitre 4 les modifications et extensions apportées au logiciel de tenue de comptes. Nous exposons ici l'évaluation que nous avons pu mener sur cette nouvelle version du programme en illustrant le rôle des outils d'appréciation développés dans le chapitre 5. C'est sur base de la démarche exposée lors du premier chapitre et déjà utilisée dans le chapitre 3 pour évaluer la première version du logiciel que nous réalisons cette évaluation.

6.1. LES OBJECTIFS DE L'EVALUATION

Ces objectifs sont identiques à ceux que nous avons déterminés pour l'évaluation de la première version du logiciel (cfr. 3.1.). Cependant, il est clair que nous portons une attention particulière aux modifications apportées au logiciel.

En effet, il est généralement assez facile de porter une critique sur l'existant (la première version du programme) aussi bien à propos des éléments de l'interface jugés incorrects et inappropriés que sur des lacunes du logiciel. La difficulté réside dans l'élaboration de solutions de remplacement, d'une part et dans la conception et le développement de nouvelles fonctionnalités, d'autre part. Il convient donc d'évaluer sérieusement tous ces changements.

6.2. LES OUTILS D'EVALUATION

Les moyens d'action disponibles lors de l'évaluation de la première version du logiciel sont encore utilisables. Certains ont subi des changements.

Le "questionnaire ergonomique" a été adapté à la nouvelle version du logiciel: nous avons ajouté des questions relatives à toute la série des éléments de l'interface que nous avons modifiée, ainsi que de nouvelles rubriques destinées à évaluer les nouvelles interfaces et les nouvelles fonctionnalités. Cette nouvelle version du questionnaire est présentée en annexe 2.

De plus, nous pouvons maintenant employer la mesure des temps pour obtenir de plus amples renseignements et de nouvelles informations ainsi que le programme d'aide à l'évaluation afin d'exploiter plus facilement les mesures prises par le programme Comptes (Chaîne des états et mesure des temps).

6.3. LA PHASE D'EXPERIMENTATION

Pour des raisons pratiques, nous n'avons pas pu mener une véritable expérimentation.

Trois individus d'une même institution ont eu l'occasion d'utiliser le logiciel au cours de trois séances. Deux de ces personnes avaient déjà utilisé la première version du logiciel et la troisième n'y avait jamais eu accès. D'autres personnes handicapées mentales ont pu essayer une seule fois le programme.

Plusieurs éducateurs nous ont par ailleurs livré leur avis sans avoir nécessairement eu l'occasion d'expérimenter le logiciel avec des personnes handicapées.

6.4. LES RESULTATS DE L'EVALUATION

Nous ne pouvons pas parler de véritables résultats mais plutôt des premières impressions de différents éducateurs.

Nous passons en revue les différents objectifs de l'évaluation même si, pour certains, rien n'a changé par rapport à la première version du logiciel.

L'UTILITE DU PROGRAMME DE TENUE DE COMPTES EN TANT QU'OUTIL D'AUTONOMIE

1) Le logiciel permet-il à une personne handicapée mentale de tenir ses comptes de manière autonome ?

Il semble que la nouvelle version du programme permette à une plus grande partie de la population handicapée de tenir ses comptes de manière autonome.

La gestion par mois facilite l'accès à une tenue des comptes autonome pour certains individus. En effet, les deux personnes qui ont utilisé la nouvelle version du logiciel et qui avaient déjà employé la précédente ont pu réellement gérer leurs comptes alors qu'auparavant le logiciel leur permettait seulement de faire des exercices de tenue de comptes. En effet, ces deux personnes sont habituées à gérer leur argent sur un cycle d'un mois et la version précédente du programme ne leur offrait pas cette possibilité.

Grâce à l'écritoire et à la calculette, le logiciel peut être utilisé entièrement avec la souris ou avec un jeu de 5 interrupteurs. Cette possibilité permet l'ouverture du logiciel à certaines personnes handicapées mentales présentant un handicap moteur associé. Désormais, il pourrait donc être un outil d'autonomie pour ce type de personnes.

2) Quelle est l'utilité du logiciel comptes ?

L'utilité pratique du logiciel ne devrait pas avoir changé sous réserve des réponses formulées ci-dessus.

3) La personne handicapée mentale éprouve-t-elle un certain plaisir à tenir ses comptes avec le logiciel ?

Les éducateurs ont apprécié une nette amélioration de l'esthétique du logiciel qui pourrait bien augmenter le plaisir qu'éprouvent certaines personnes car elles semblent également accorder beaucoup d'importance à ce critère.

4) Le logiciel peut-il inciter des personnes handicapées mentales à tenir leurs comptes ?

Il conviendrait de mesurer (mais comment ?) si l'ensemble des modifications apportées au logiciel pourrait avoir une influence par rapport à cet objectif.

L'UTILITE DU PROGRAMME EN TANT QU'OUTIL D'APPRENTISSAGE

Les objectifs 5 à 9 concernent les apprentissages des individus (termes, concepts, acquis, comportements et notion des valeurs). Très peu de changements sont attendus pour ces objectifs. Néanmoins, l'utilisation des réglettes pourrait avoir un effet positif à long terme sur l'acquisition de la notion des valeurs. En effet, les réglettes font correspondre une longueur à chaque montant. Nous pourrions faire l'hypothèse que l'utilisateur, à force d'être amené à comparer ces différentes longueurs entre elles, pourra comparer des montants entre eux. C'est d'ailleurs une des idées de base sur laquelle se fondent les réglettes de Cuisenaire.

L'INTERFACE ET LES FONCTIONNALITES DU LOGICIEL

10) Les fonctionnalites offertes par le logiciel sont-elles suffisantes et satisfaisantes?

La tenue des comptes sur un cycle mensuel est une fonctionnalité qui a été très demandée et les éducateurs semblent très satisfaits de cette réalisation.

La fonction de correction du porte-monnaie est considérée comme intéressante mais n'a pas encore été utilisée dans la pratique : ceci est assez normal car il s'agit d'une "solution de secours" qui ne doit être utilisée que lorsqu'il y a eu une erreur.

11) L'interface réalisée dans le logiciel convient-elle bien aux personnes handicapées mentales ou devrait-elle être amélioree ?

L'écritoire qui permet à une personne d'écrire son nom à l'aide de la souris n'a pas encore été utilisé et nous ne pouvons donc encore rien dire sur la façon dont est comprise cette interface mais, de l'avis des éducateurs, elle présente un intérêt pour certains individus. Cependant, Monsieur Baechler nous a appris que la majorité des claviers suisses sont de type QWERTY. Il serait donc nécessaire d'adapter notre écritoire pour la version suisse du logiciel.

La calculette, par contre, a été employée par les trois individus et a été bien comprise par ceux-ci. Cette possibilité offerte pour entrer les montants d'argent a été jugée "très intéressante" par l'éducateur.

Les réglettes n'ont pas été immédiatement bien comprises par les trois individus. Cependant, selon l'éducateur, elles peuvent être rapidement maîtrisées à condition de mettre en oeuvre un travail préparatoire avec des individus avant de commencer l'expérimentation. Les autres éducateurs croient également en leurs chances de succès.

Le terme "OK" qui remplace "Fini" dans de nombreux écrans semble plus adéquat et est compris sans difficulté par les utilisateurs.

Les icônes de billets pour la saisie d'une somme par "billets" sont maintenant dessinées dans une couleur qui se rapproche autant que possible de la réalité et cela est tout à fait utile pour les trois personnes handicapées. De plus, le chevauchement des billets (feed-back) qui se présentent dans la partie droite de l'écran a été jugé meilleur qu'auparavant.

Nous avions supprimé la balance car elle était fausse et pas très bien comprise comme nous l'avons vu au point 3.5 mais il semble que sa symbolique aide pourtant certains individus. Il faudrait donc sans doute la modifier et ensuite la réintégrer au programme.

Nous n'avons pas pu récolter davantage d'informations sur les autres modifications que nous avons apportées au logiciel.

6.5. CRITIQUE DES OUTILS D'EVALUATION UTILISES

Nous ne pouvons évidemment pas encore tirer beaucoup de conclusions sur les nouveaux outils. Nous avons cependant pu constater que les mesures de temps pourraient nous aider à déterminer la durée d'apprentissage. Nous avons d'ailleurs utilisé le programme d'aide à l'évaluation pour observer les mesures de temps et les présenter graphiquement : nous avons remarqué très nettement une tendance des temps à diminuer de séance en séance.

Le graphique présenté à la section 5.3 a d'ailleurs été établi par le programme de dépouillement à partir d'une des trois expérimentations menées avec la nouvelle version du logiciel Comptes.

Si l'outil est très utile pour les évaluateurs, il semble également intéresser les éducateurs.

CHAPITRE VII : ESQUISSE D'UNE AUTRE APPROCHE POUR EVALUER UN LOGICIEL POUR PERSONNES HANDICAPEES MENTALES

"La puissance d'une recherche, c'est-à-dire sa capacité à établir avec une grande certitude la confirmation des hypothèses, va dépendre de plusieurs facteurs dont notamment la qualité des observations et des mesures ainsi que celle des instruments qui les produisent." J.P. Beaugrand, dans Fondements et étapes de la recherche scientifique en psychologie, 1982.

Dans le chapitre 3, nous avons défini les conditions d'expérimentation utilisées pour évaluer le logiciel Comptes et nous avons relevé certaines faiblesses, limites et difficultés liées à ce mode d'expérimentation.

Nous proposons ici de lancer les bases d'une autre modalité d'expérimentation. Pour ce faire, nous verrons :
- les nouvelles conditions d'expérimentation que nous préconisons;
- les nouveaux outils d'évaluation imaginés et l'adaptation des anciens outils à ce nouveau mode d'expérimentation;
- l'application des mathématiques floues à ces outils pour l'évaluation de logiciels.

7.1 UNE ALTERNATIVE AUX CONDITIONS D'EXPERIMENTATION UTILISEES

On constitue une équipe de juges qui a pour mission d'évaluer le logiciel ou plus précisément d'apporter des éléments de réponse aux objectifs de l'évaluation.

Afin de remplir cette tâche, les juges définiront les caractéristiques de l'échantillon c'est-à-dire le nombre d'individus, leur degré d'incapacité, le type d'institution dont ils font partie,...

Ils choisiront et imagineront également les outils d'évaluation dont ils se serviront. En outre, ils préciserons le contenu de chaque outil. Ils définiront ainsi, par exemple, l'ensemble des questions constituant leur questionnaire.

La particularité de la nouvelle modalité d'expérimentation peut se résumer ainsi : tous les juges rencontreront ensemble toutes les personnes handicapées mentales choisies pour l'expérimentation, assisteront à quelques séances d'utilisation du logiciel et porteront chacun un jugement autonome sur l'interaction entre l'utilisateur et le logiciel. Commentons à présent ces différentes exigences.

En fait, tous les juges doivent disposer des mêmes observations afin d'émettre leur jugement. Ces mêmes "conditions d'observations" impliquent qu'ils doivent rencontrer absolument tous les utilisateurs et qu'ils doivent être présents en même temps lors de ces différentes rencontres. Le nombre de juges constituant l'équipe sera donc fixé avant toute expérimentation et ne pourra pas changer au cours de l'expérimentation. Notons qu'il pourrait éventuellement se ramener à une seule personne.

Ces juges ne devront évidemment pas assister à toutes les séances d'utilisation du logiciel par la personne handicapée. En fonction des objectifs d'évaluation qu'ils se sont fixés, de la qualité des résultats qu'ils recherchent et du type de logiciel, ils détermineront le nombre de séances auxquelles ils assisteront et le nombre de séances séparant chacune de leurs visites.

Au cours des séances auxquelles ils assisteront, chacun des juges évaluera l'utilisation du programme par la personne handicapée mentale avec le ou les outils d'appréciation disponibles.

Il est important de veiller à ce que chacun évalue de manière autonome malgré des conditions d'observations identiques: on s'arrangera pour qu'ils se concertent le moins possible. Si l'équipe est constituée de n experts, il est effectivement plus riche d'avoir n jugements sur le même fait observé au cours d'une expérimentation que d'avoir une information moyenne résumant les avis de ces n experts.

Pour rendre opérationnelles ces exigences, il conviendra de choisir des juges qui ont une assez grande disponibilité pour l'évaluation et qui pourront se rendre sur les différents sites d'expérimentation ensemble.

Pour l'évaluation du logiciel Comptes, l'équipe pourrait, par exemple, être constituée d'un étudiant terminal en psychologie et d'un étudiant en informatique sensibilisés au milieu du handicap mental.

Ces personnes que nous appelons juges devront se familiariser avec les personnes handicapées mentales qui constituent l'échantillon. Ainsi, il serait bon que des contacts préalables à toute expérimentation puissent s'établir à plusieurs reprises en dehors du contexte même du logiciel. Pour l'évaluation du logiciel Comptes, il serait intéressant que les juges partagent des activités familières aux personnes handicapées mentales et puissent, par exemple, aller faire les courses avec elles.

Lors de l'expérimentation, les juges ne doivent pas donner l'impression aux utilisateurs que c'est eux qu'ils évaluent. Pratiquement, ils devront tenter de prendre un strict minimum de notes, de paraître naturels et de créer une ambiance détendue. Ils ne coteront qu'en l'absence de la personne.

Toutes ces exigences ne seront peut-être pas évidentes à satisfaire mais elles nous permettront de développer des outils d'évaluation qui sont susceptibles de nous fournir des résultats plus fiables et plus précis que ceux que nous avons pu obtenir.

7.2. LE DEVELOPPEMENT DE NOUVEAUX OUTILS D'EVALUATION UTILISABLES AVEC LES CONDITIONS D'EXPERIMENTATION

Nous présentons à présent deux outils d'évaluation qui s'adaptent bien aux nouvelles conditions d'expérimentation que nous venons d'exposer.

7.2.1. LE QUESTIONNAIRE "VALUE"

La réponse fournie par un juge à une quelconque question fermée ne doit plus nécessairement être binaire. Puisque le même juge évalue l'ensemble des utilisateurs sur base d'un même référentiel, il pourra, pour chaque élément de ce référentiel et pour chaque utilisateur, donner une note valuée et donc plus nuancée.

Le problème est de déterminer, pour chaque question, combien de nuances il convient d'offrir aux juges .

Il est souhaitable, pour des raisons de dépouillement, de définir un nombre impair de nuances.

Un nombre trop faible de choix de réponses ne leur laisserait pas assez de latitude alors qu'un nombre trop élevé est inutile et les laisserait souvent hésitants quant à la nuance à choisir. Il semblerait qu'un nombre de 11 graduations soit optimal. Ceci a été vérifié par un grand nombre d'expériences dans divers domaines. De plus, on peut ainsi rester dans le système décimal.

Par conséquent, on demandera aux juges de répondre pour chaque question par un entier compris entre 0 et 10 (bornes incluses) ou par un nombre compris entre 0 et 1 (bornes incluses) et ne possédant qu'une seule décimale.

Afin que les personnes puissent répondre aux questions, il convient d'associer à chacune des 11 positions une sémantique définie le plus précisément possible. On construit donc une grille de correspondance numérique/sémantique appelée aussi échelle d'appréciation.

- On pourra utiliser une échelle de type "Likert" comme celle proposée par Arnold Kaufmann :

REPONSE NUMERIQUE	EQUIVALENT SEMANTIQUE
1	Vrai
0.9	Quasiment vrai
0.8	Presque vrai
0.7	Plutôt vrai
0.6	Plus vrai que faux
0.5	Ni vrai ni faux (indécidable)
0.4	Plus faux que vrai
0.3	Plutôt faux
0.2	Presque faux
0.1	Quasiment faux
0	Faux

Les questions doivent être exprimées sous la forme de propositions positives, affirmatives et neutres.

La réponse doit être comprise comme le degré de certitude que le juge accorde à la proposition. Par exemple, une réponse de 0.7 à la question "La personne comprend le message Y." doit être interprétée de cette façon : "le juge pense qu'il est plutôt vrai (0.7) de dire que la personne comprend le message Y".

On pourrait également songer à utiliser un autre type d'échelle comme celle que nous proposons ci-dessous :

REPONSE NUMERIQUE	EQUIVALENT SEMANTIQUE
1	Tout à fait
0.9	Très bien
0.8	Bien
0.7	Assez bien
0.6	Plutôt bien que mal
0.5	A moitié
0.4	Plutôt mal que bien
0.3	Assez mal
0.2	Mal
0.1	Très mal
0	Pas du tout

Les résultats obtenus sur ce type de grille peuvent être facilement exploités par des méthodes statistiques.

Dans le premier cas, la cote reflète un degré d'accord avec une proposition à partir d'observations.

La seconde grille est plutôt du type qualitatif puisque les juges apprécient de quelle manière la personne interagit avec le programme, comprend des termes, se débrouille pour tenir ses comptes,...

Dans le cadre des conditions d'expérimentation que nous avons utilisées, chaque éducateur prenait en charge un petit groupe de personnes handicapées mentales, c'est-à-dire une petite partie de l'échantillon. Dans de telles circonstances, il est plus difficile de permettre aux éducateurs de répondre aux questions par une "cote nuancée" car on risque d'observer des grandes variations dans la façon de répondre d'un éducateur à l'autre. Par contre, on peut espérer qu'une seule personne reste cohérente avec elle-même. Sa subjectivité interviendra bien entendu dans les réponses mais de la même façon pour l'ensemble de l'échantillon.

Par ailleurs, afin de ne pas dépendre de la subjectivité d'une seule et même personne, il est préférable que plusieurs personnes donnent leur avis.

Les nouvelles conditions d'expérimentation devraient nous donner des informations plus fiables et plus précises.

7.2.2. LE JEU DE ROLE

Il s'agit de "jouer" avec la personne handicapée mentale à faire des courses et à tenir ses comptes. Une fois entré dans le jeu, on est dans la "réalité". Voyons maintenant la manière dont pourrait se dérouler un tel jeu, les moments auxquels on "jouerait" et l'intérêt d'un tel jeu pour l'évaluation d'un logiciel. Nous prenons ici l'exemple du logiciel de tenue des comptes.

Un groupe de juges donnerait au début du jeu de l'argent à la personne handicapée mentale et celle-ci pourrait acheter à son gré des articles réels que les juges auraient apportés et qui constitueraient le magasin. Un des juges jouerait le rôle du magasinier, l'autre pourrait, en cours de jeu, offrir de l'argent supplémentaire afin que la personne puisse éventuellement faire d'autres achats. Ensuite, on demanderait à la personne handicapée mentale de faire ses comptes c'est-à-dire d'énumérer ses recettes, ses dépenses, l'état de son porte-monnaie et de vérifier si ses comptes sont justes. Le jeu prendrait alors fin. Remarquons que le jeu se déroule sans que l'on se soit servi du logiciel.

Ce jeu pourrait avoir lieu avant la toute première utilisation du programme Comptes ce qui permettrait aux juges d'évaluer les acquis de la

personne handicapée en ce qui concerne le calcul, la notion du prix des choses, la capacité à tenir ses comptes dans la réalité...

On rejouerait avec la personne handicapée à la fin de l'expérimentation du logiciel. A ce moment, on pourrait alors évaluer d'éventuels apprentissages faits par la personne handicapée mentale grâce au logiciel. Ces apprentissages portent sur la capacité à tenir ses comptes, à distinguer les recettes des dépenses,... dans la réalité sans l'aide directe du logiciel.

Enfin, la personne passerait à l'ordinateur, elle lui communiquerait le contenu du porte-monnaie, l'ensemble de ses recettes et dépenses et vérifierait ses comptes cette fois-ci avec l'aide de l'ordinateur.

Par ce moyen, on essaye d'évaluer les avantages et inconvénients du logiciel comme outil de tenue de comptes.
On pourrait ainsi vérifier si:
- la personne sait distinguer ses recettes de ses dépenses;
- la personne connaît l'état de son porte-monnaie;
- la personne peut vérifier d'elle-même que ses comptes sont justes;
- la personne peut vérifier sur l'ordinateur que ses comptes sont justes.

Pour rendre opérationnel cet outil d'appréciation, il faudra avant toute expérimentation que les juges définissent précisément le déroulement du jeu, les moments où le jeu se déroulera, les objectifs de l'évaluation auxquels ils veulent apporter une réponse grâce à cet outil. A partir de ces objectifs, ils définiront un référentiel commun pour l'évaluation et le jeu pourra se dérouler dans des conditions semblables pour tous les individus constituant l'échantillon. Pour chaque élément du référentiel, ils pourront apporter une réponse nuancée en utilisant, par exemple, une des deux grilles définies au point 7.2.1. Pour l'évaluation, les informations récoltées de cette manière seront donc plus fiables et plus précises.

Après avoir défini et mis en évidence les limites et les difficultés des conditions d'expérimentation utilisées pour évaluer le programme Comptes, nous avons proposé une forme d'expérimentation différente, nous avons enfin montré l'adaptation du questionnaire à cette nouvelle expérimentation et lancé les bases d'un autre outil d'évaluation: le jeu de rôle.

Pour obtenir des résultats concrets par rapport aux objectifs de l'évaluation, nous devons à présent disposer d'une méthode de dépouillement des résultats.

Nous exposons donc des éléments de la théorie des mathématiques floues et ses applications à cette nouvelle forme d'expérimentation mais ce n'est bien sûr pas la seule méthode utilisable et disponible. L'analyse en composantes principales peut, par exemple, elle aussi être utilisée dans cette optique. De nombreux autres modèles statistiques pourraient également s'appliquer aux données récoltées de cette manière.
Les mathématiques floues ne sont donc ici qu'une méthode de dépouillement parmi d'autres.

7.3 APPLICATION DES MATHEMATIQUES FLOUES AUX OUTILS D'EVALUATION.

"Plus on creuse la science, plus elle s'élève. Nous pouvons donc être certains que la multiplication des méthodes, à quelque étage que ces méthodes travaillent, ne saurait nuire à l'unité de la science"

G. Bachelard.

Notre objectif est plus de montrer les applications possibles des mathématiques floues à l'évaluation d'un logiciel que d'exposer la théorie mathématique en elle-même. Nous nous basons sur l'exposé de A. Kaufmann présenté dans le cadre du stage de formation "Application des concepts et outils issus des mathématiques floues aux problèmes d'évaluation dans les sciences humaines et sociales", qui a eu lieu les 10, 11, 12 et 13 juillet 1990 à l'Université de Technologie de Compiègne.

Considérons p experts* chargés d'évaluer le logiciel. Chaque expert a répondu pour les m utilisateurs du logiciel à l'ensemble du référentiel composé par exemple de n questions.

Notons:

Q_i : la $i^{ème}$ question (avec $1 \leq i \leq n$) où n vaut 4 dans notre exemple;

Ind_j : le $j^{ème}$ individu (avec $1 \leq j \leq m$) où m vaut 3 pour notre exemple;

E_k : le $k^{ème}$ expert (avec $1 \leq k \leq p$) où p vaut 3 dans notre exemple;

X_{ijk} : la réponse du $k^{ème}$ expert à la $i^{ème}$ question pour le $j^{ème}$ individu.

- Imaginons que notre questionnaire soit composé des 4 questions suivantes:

Q_1 : La personne sait lire
Q_2 : La personne sait compter
Q_3 : Le son est utile à la personne
Q_4 : La personne comprend la signification des couleurs.

et supposons que nous ayons obtenu les résultats suivants:

Expert E_1	Ind_1	Ind_2	Ind_3
Q_1	0.7	0.2	0.5
Q_2	0.5	0.2	0.4
Q_3	0.2	0.6	0.5
Q_4	0.9	0.3	0.4

---

* Dans la théorie des mathématiques floues, le terme "expert" correspond à celui de "juge" dont nous parlions ci-dessus.

Expert E_2	Ind_1	Ind_2	Ind_3
Q_1	0.6	0.3	0.4
Q_2	0.5	0.1	0.4
Q_3	0.4	0.2	0.6
Q_4	0.5	0.2	0.3

Expert E_3	Ind_1	Ind_2	Ind_3
Q_1	0.5	0.3	0.4
Q_2	0.4	0	0.3
Q_3	0.4	0.2	0.6
Q_4	0.4	0.1	0.3

L'interprétation doit se faire avec la grille de correspondance de type "Likert" définie à la section 7.2.1. :

L'expert E_1 a ainsi jugé qu'il est quasiment vrai (0.9) de dire que l'individu 1 comprend la signification des couleurs (Q_4).

Il est intéressant avant d'aller plus loin de mesurer les différences de jugement entre experts. L'écart entre deux experts peut être calculé par la distance de Hamming qui vaut la somme des valeurs absolues des différences. Cette distance sera divisée par le nombre d'observations de manière à obtenir un résultat toujours compris entre 0 et 1.

La distance totale entre deux experts (E_k et E_l) est donnée par la formule :

$$d(E_k, E_l) = \frac{\sum_{i=1}^{n} \left(\frac{\sum_{j=1}^{m} | X_{ijk} - X_{ijl} |}{m} \right)}{n}$$

L'écart de jugement entre deux experts (E_k et E_l) sur la question Q_i vaut :

$$d_{Q_i}(E_k, E_l) = \frac{\sum_{j=1}^{m} \left| X_{ijk} - X_{ijl} \right|}{m}$$

L'écart de jugement entre les deux experts E_k et E_l à propos de l'individu Ind_j vaut :

$$d_{Ind_j}(E_k, E_l) = \frac{\sum_{i=1}^{n} \left| X_{ijk} - X_{ijl} \right|}{n}$$

Revenons à notre exemple où l'écart de jugement entre E_2 et E_3 sur la question Q_2 vaut :

$$d_{Q_2}(E_2, E_3) = \frac{|0.5-0.4| + |0.1-0| + |0.4-0.3|}{3}$$

$$= \frac{0.3}{3} = 0.1$$

L'écart de jugement entre E_1 et E_3 sur l'individu Ind_1 vaut:

$$d_{Ind_j}(E_k, E_l) = \frac{|0.7-0.5| + |0.5-0.4| + |0.2-0.4|}{4}$$

$$= \frac{1.0}{4} = 0.25$$

L'écart de jugement entre E_1 et E_2 vaut :

$$d(E_1, E_2) = (|0.7-0.6| + |0.5-0.5| + |0.2-0.4| + |0.9-0.5|$$
$$+ |0.2-0.3| + |0.2-0.1| + |0.6-0.2| + |0.3-0.2|$$
$$+ |0.5-0.4| + |0.4-0.4| + |0.5-0.6| + |0.4-0.3|)/2$$

$$= 1.7/12 = 0.14167$$

De la même manière, il vient :

$d(E_2,E_3) = 0.6 / 12 = 0.05$
$d(E_1,E_3) = 2.3 / 12 = 0.1917$

A partir de ces distances totales entre experts, on peut définir la matrice symétrique de dissemblance D dont les éléments d_{ij} sont définis :

$d_{ij} = d(E_i,E_j)$ si i est différent de j
$= 0$ s'ils sont égaux

	E_1	E_2	E_3
E_1	0	0.14167	0.1917
E_2	0.14167	0	0.05
E_3	0.1917	0.05	0

Plus l'écart entre deux experts est faible, plus leur expertise est proche, semblable.
Ainsi les expertises rendues par E_2 et E_3 sont plus proches entre elles que celles rendues par E_1 et E_3.

Si les écarts mesurés entre deux experts sont trop importants*, les experts doivent se concerter sur les raisons de cette discordance d'expertise.

Nous allons maintenant calculer un experton qui nous permettra de tirer des conclusions sur l'évaluation du logiciel. Un experton est un tableau reprenant l'avis des différents experts sur chaque élément du référentiel.

Dans le cas de l'évaluation du logiciel grâce à un questionnaire, le référentiel est double. Il est bien entendu constitué de l'ensemble des questions mais aussi de l'ensemble des individus de l'échantillon.

Calcul des expertons

Individus:	Ind1				Ind2				Ind3			
questions:	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
experts E1	.7	.5	.2	.9	.2	.2	.6	.3	.5	.4	.5	.4
E2	.6	.5	.4	.5	.3	.1	.3	.2	.4	.4	.6	.3
E3	.5	.4	.4	.4	.3	0	.3	.1	.4	.3	.6	.3

Statistique sur les bornes

niveaux												
0	0	0	0	0	0	1	0	0	0	0	0	0
.1	0	0	0	0	0	1	0	1	0	0	0	0
.2	0	0	1	0	1	1	0	1	0	0	0	0
.3	0	0	0	0	2	0	2	1	0	1	0	2
.4	0	1	2	1	0	0	0	0	2	2	0	1
.5	1	2	0	1	0	0	0	0	1	0	1	0
.6	1	0	0	0	0	0	1	0	0	0	2	0
.7	1	0	0	0	0	0	0	0	0	0	0	0
.8	0	0	0	0	0	0	0	0	0	0	0	0
.9	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0

---

* A. Kaufmann estime que pour ce genre de problèmes, un écart est jugé trop important s'il est supérieur à 0.20 .

Diviser par le nombre d'experts (c'est-à-dire 3)

division par le nombre d'experts:

niveaux												
0	0	0	0	0	0	1/3	0	0	0	0	0	0
.1	0	0	0	0	0	1/3	0	1/3	0	0	0	0
.2	0	0	1/3	0	1/3	1/3	0	1/3	0	0	0	0
.3	0	0	0	0	2/3	0	2/3	1/3	0	1/3	0	2/3
.4	0	1/3	2/3	1/3	0	0	0	0	2/3	2/3	0	1/3
.5	1/3	2/3	0	1/3	0	0	0	0	1/3	0	1/3	0
.6	1/3	0	0	0	0	0	1/3	0	0	0	2/3	0
.7	1/3	0	0	0	0	0	0	0	0	0	0	0
.8	0	0	0	0	0	0	0	0	0	0	0	0
.9	0	.0	0	1/3	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0

Cumuls

Afin d'obtenir une décroissance monotone, on cumule les fréquences en partant du niveau inférieur. On obtient ainsi l'experton suivant :

	Ind1				Ind2				Ind3			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
niveaux												
0	1	1	1	1	1	1	1	1	1	1	1	1
.1	1	1	1	1	1	2/3	1	1	1	1	1	1
.2	1	1	1	1	1	1/3	1	2/3	1	1	1	1
.3	1	1	2/3	1	2/3	0	1	1/3	1	1	1	1
.4	1	1	2/3	1	0	0	1/3	0	1	2/3	1	1/3
.5	1	2/3	0	2/3	0	0	1/3	0	1/3	0	1	0
.6	2/3	0	0	1/3	0	0	1/3	0	0	0	2/3	0
.7	1/3	0	0	1/3	0	0	0	0	0	0	0	0
.8	0	0	0	1/3	0	0	0	0	0	0	0	0
.9	0	0	0	1/3	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0

Par construction, la ligne correspondant au niveau 0 est toujours constituée de 1 et ne sera donc pas prise en compte dans les calculs suivants.

Toute une série de résultats peuvent être tirés de ce tableau d'expertons.

Notons X_{jiv} le résultat au $v^{ème}$ niveau de l'experton correspondant à le $i^{ème}$ question pour le $j^{ème}$ individu.

1) Comparaison des individus :

On calcule pour une question Q_i, pour chaque individu Indj, la valeur

$$\sum_{v=0.1}^{1} X_{jiv}$$

Exemple :

Ind_1 :

$$\frac{\sum_{V=0.1}^{1} X_{11V}}{10} = \frac{1.8}{3}$$

Ind_2 : $\frac{0.8}{3}$

Ind_3 : $\frac{1.3}{3}$

Parmi les 3 individus, c'est l'individu 1 qui sait le mieux lire (Q_1) des trois.

On peut obtenir ainsi un ordre ou classement des utilisateurs par rapport à une question.

2) Evaluation des prérequis de l'échantillon :

On peut, en appliquant la même formule que ci-dessus, obtenir des résultats sur les acquis des membres de l'échantillon. Exemple:

Ind1 :

$$\frac{\sum_{V=0.1}^{1} X_{12V}}{10} = \frac{1.4}{3}$$

Ind2 : $\frac{0.3}{3}$

Ind3 : $\frac{1.1}{3}$

Aucun utilisateur n'atteint de score supérieur ou égal à 0.6.

Il apparaît ainsi dans notre cas, qu'aucun utilisateur ne sait compter (Q_2).

3) Evaluation de l'interface :

On peut, toujours avec la même formule, ne plus évaluer les utilisateurs, mais l'utilité ou la qualité d'une interface, en ne travaillant plus sur une question du questionnaire psychologique, mais sur une question ayant trait à l'ergonomie du programme.

4) Détermination de groupes d'individus qui correspondent à plusieurs critères :

Il est souvent intéressant de repérer des individus ou des groupes d'individus qui répondent à plus d'un seul critère. On fait alors des croisements entre plusieurs questions. Ainsi, on peut se demander quels sont les utilisateurs qui savent lire (Q1) et qui comprennent la signification des couleurs (Q4).(1).

Pour chaque individu j, on calcule un experton $\text{Min}(X_{j1v}, X_{j4v})$. On calcule ensuite la moyenne de ces expertons, ce qui correspond à :

$$\frac{\sum_{v=0.1}^{1} \min(X_{j1v}, X_{j4v})}{10}$$

On obtient donc une moyenne par utilisateur pour la conjonction entre la question 1 et la question 4. C'est la moyenne d'un experton représentant la conjonction entre deux questions, et non pas la conjonction des moyennes des expertons relatifs à chacune des deux questions. Il est important de calculer la moyenne à la fin des calculs de manière à faire tomber l'entropie le plus tard possible et à travailler ainsi le plus longtemps possible avec des informations plus riches.

---

(1) Remarque : Pour traduire la conjonction logique "a et b", nous avons choisi de prendre l'opérateur Min(a,b).

D'autres choix auraient été possible. On aurait par exemple pu prendre aussi :

$$\frac{a\ b}{1+\overline{a}.\overline{b}} \quad \text{avec } \overline{a} = 1-a \text{ et } b = 1-\overline{b} \quad \text{OU} \quad 0 \vee (a+b-1)$$

Prendre l'opérateur Min revient à choisir le cas le plus défavorable.

individus:	Ind1	Ind2	Ind3
questions:	Q1 ^ Q4	Q1 ^ Q4	Q1 ^ Q4
niveaux			
0	1	1	1
.1	1	1	1
.2	1	2/3	1
.3	1	1/3	1
.4	1	0	1/3
.5	2/3	0	0
.6	1/3	0	0
.7	1/3	0	0
.8	0	0	0
.9	0	0	0
1	0	0	0
moyenne	1.6/3	0.6/3	1/3

On voit donc qu'il est plus vrai que faux (0.6) de dire que l'individu 1 sait lire et comprend la signification des couleurs.

Ce genre de croisement entre questions est évidemment d'une grande utilité pour l'évaluation des capacités et acquis de la personne handicapée mentale mais ces croisements peuvent aussi être utiles pour juger de la pertinence, de l'utilité d'un type d'interface par rapport à un type d'utilisateur.

Exemple : Y a-t-il des utilisateurs qui ne savent ni lire (non Q_1), ni compter (non Q_2) et qui ne comprennent pas la signification des couleurs (non Q4) ?

Le calcul se déroule ici en trois étapes :

1. On calcule d'abord, pour chaque utilisateur, les expertons complémentaires (pour nier la question) de chacune des trois questions Q1, Q2 et Q4 pour traduire la négation.

2. Ensuite, pour chaque utilisateur, on calcule l'experton minimum des trois expertons obtenus en 1 pour traduire la conjonction.
3. A la fin seulement, on calcule, pour chaque utilisateur, la moyenne de l'experton obtenu en 2.

1ère étape : Calcul des expertons complémentaires :

individus		Ind1			Ind2			Ind3	
questions	non Q1	nonQ2	nonQ4	nonQ1	nonQ2	nonQ4	nonQ1	nonQ2	nonQ4
0	1	1	1	1	1	1	1	1	1
.1	1	1	1	1	1	1	1	1	1
.2	1	1	2/3	1	1	1	1	1	1
.3	1	1	2/3	1	1	1	1	1	1
.4	2/3	1	2/3	1	1	1	1	1	1
.5	1/3	1	2/3	1	1	1	1	1	1
.6	0	1/3	1/3	1	1	1	2/3	1	1
.7	0	0	0	1	1	1	0	1/3	2/3
.8	0	0	0	1/3	1	2/3	0	0	0
.9	0	0	0	0	2/3	1/3	0	0	0
1	0	0	0	0	1/3	0	0	0	0

2ème étape : calcul des minima

individus	Ind1			Ind2			Ind3		
questions	min de			min de			min de		
	non Q1	non Q2	non Q4	non Q1	non Q2	non Q4	non Q1	non Q2	non Q4
niveaux									
0	1			1			1		
.1	1			1			1		
.2	2/3			1			1		
.3	2/3			1			1		
.4	2/3			1			1		
.5	1/3			1			1		
.6	0			1			2/3		
.7	0			1			0		
.8	0			1/3			0		
.9	0			0			0		
1	0			0			0		
3ème étape: Calcul des moyennes									
	1/3 = 0.333			2.2/3 = 0.733			1.7/3 = 0.566		

Il est plutôt vrai (0.733) de dire que l'individu Ind2 ne sait ni lire ni écrire et ne comprend pas la signification des couleurs. Il est plutôt faux (0.333) de dire que l'individu Ind1 ne sait ni lire ni écrire et ne comprend pas la significations des couleurs.

On peut aussi s'intéresser à repérer des individus satisfaisant à un critère ou à un autre. En général, le "ou" exclusif sera traduit par un opérateur Max, mais on peut aussi en utiliser d'autres, comme par exemple:

$$\frac{a+b}{1+a.b} \quad \text{ou} \quad 1 \wedge (a+b)$$

Il est possible de combiner tous les opérateurs ensemble. On pourrait ainsi chercher les personnes pour qui il est vrai de dire qu'elles savent lire (Q1) ou compter (Q2) et pour qui le son n'est pas utile (Q4).
Pour chaque utilisateur, le calcul se déroule ici en quatre étapes:

1) calcul de l'experton qui correspond à non Q4
2) calcul de l'experton qui correspond au max(Q1,Q2)
3) calcul de l'experton qui correspond au min entre l'experton obtenu en 1 et celui obtenu en 2.
4) calcul de la moyenne de l'experton obtenu en 3.

5) Calcul de la distance entre individus :

La distance de Hamming entre deux individus j et j' sur l'ensemble des n questions vaut :

$$d(j,j') = \frac{\sum_{i=1}^{n} \sum_{v=0}^{1} \left| X_{jiv} - X_{j'iv} \right|}{10 * n}$$

Il vient ainsi :

d (Ind1,Ind2) = 3.7/12 = 0.3083
d (Ind_1,Ind_3) = 0.19166
d (Ind_2,Ind_3) = 0.1833

On peut, dès lors, construire la matrice symétrique de dissemblance entre individus :

	Ind_1	Ind_2	Ind_3
Ind_1	0	0.3083	0.1916
Ind_2	0.3083	0	0.1833
Ind_3	0.1916	0.1833	0

Si on prend le complément à 1 des valeurs dans les cases de cette relation de distances, on obtient une relation de ressemblance. En effet, plus la distance entre individus est faible, plus on peut admettre qu'ils se ressemblent.

	Ind_1	Ind_2	Ind_3
Ind_1	1	0.6917	0.8083
Ind_2	0.6917	1	0.8167
Ind_3	0.8083	0.8167	1

Au niveau $\alpha = 1$, tous les individus sont distincts :

$\alpha=1$	Ind_1	Ind_2	Ind_3
Ind_1	1	0	0
Ind_2	0	1	0
Ind_3	0	0	1

On obtient trois relations maximales de similitudes : {1}, {2}, {3}.

Au niveau $\alpha \geq 0.8166$:

$\alpha \geq 0.8166$	Ind_1	Ind_2	Ind_3
Ind_1	1	0	0
Ind_2	0	1	1
Ind_3	0	1	1

A ce niveau, les individus Ind_2 et Ind_3 sont proches. On obtient deux relations maximales de similitudes {2,3} et {1}.

Au niveau $\alpha \geq 0.8083$:

$\alpha \geq 0.8083$	Ind_1	Ind_2	Ind_3
Ind_1	1	0	1
Ind_2	0	1	1
Ind_3	1	1	1

A ce niveau, l'individu Ind_1 est proche de Ind_2, ce deuxième est proche de Ind_3 mais Ind_1 n'est pas proche de Ind_3.

On obtient donc deux relations maximales de similitude : {1,3} et {2,3}.

Au niveau $\alpha \geq 0.6917$

$\alpha \geq 0.6917$	Ind_1	Ind_2	Ind_3
Ind_1	1	1	1
Ind_2	1	1	1
Ind_3	1	1	1

On a une seule relation maximale de similitude : {1,2,3}.

Ces calculs peuvent être très utiles pour construire des groupes homogènes d'utilisateurs du logiciel.

6°) Calcul de la distance entre questions.

La distance de Hamming entre deux questions Q_i et $Q_{i'}$ pour m individus est définie comme suit :

$$d(Q_i, Q_{i'}) = \frac{\sum_{j=1}^{m} \sum_{v=0,1} \left| X_{jiv} - X_{ji'v} \right|}{(10 * m)}$$

On peut ainsi calculer la distance entre toutes les questions prises deux à deux puis construire la matrice de dissemblance entre questions.
Le complément de celle-ci donnera la relation de ressemblance à partir de laquelle on pourra calculer les sous-relations maximales de similitude à différents niveaux α.

La démarche est donc identique à ce qui a été fait pour les individus.

Ici, on ne cherche plus à dégager des groupes homogènes d'utilisateurs mais à mettre en évidence les questions les plus redondantes et les plus discriminantes.

En fait, la recherche des sous-relations maximales de similitude joue dans les mathématiques floues le rôle de la recherche de classes d'équivalence dans la théorie classique (matrice de corrélation).

D'après Arnold Kaufmann, l'avantage de l'utilisation des mathématiques floues par rapport aux méthodes statistiques classiques peut se résumer comme suit.
Pour transférer les opinions, c'est-à-dire les données récoltées par les outils d'appréciation avec les nouvelles conditions d'expérimentation, en résultats par rapport aux objectifs de l'évaluation, nous devons mettre en place des opérateurs. Un opérateur comme l'espérance mathématique implique la linéarité. Or, les raisonnements logiques, eux, ne sont pas linéaires. Les raisonnements basés sur le calcul des expertons permettent de conserver toute la précision des différentes opinions au cours de l'application des différents opérateurs. La moyenne n'est calculée qu'à la fin des opérations pour faire tomber l'entropie le plus tard possible.

Remarques :

1) Dans le but d'évaluer les apprentissages faits par la personne handicapée mentale en ce qui concerne les notions, les concepts ou les nouveaux comportements, il est bien sûr possible de comparer des expertises réalisées après différentes séances d'utilisation du logiciel. Ces comparaisons se feront grâce à la distance de Hamming calculée encore entre expertons.

2) Il est possible de ne pas demander aux experts de coter par une seule note, mais par un intervalle de la forme [a1, a2] où $a1 \leq a2$ et a1, a2 sont des valuations. En effet, dans la pratique, l'expert aura parfois des difficultés à répondre avec une seule valeur à une question.

Tout ce que nous avons expliqué ci-dessus reste valable en tenant compte des propriétés suivantes :

. ([a1 a2] < [b1 b2]) => ($a1 \geq b1, b2 \geq a2$)
. Le complémentaire de l'intervalle [a1 a2] vaut [1-a2 1-a1]
. La moyenne de l'intervalle [a1 a2] vaut (a1+a2) / 2
. La distance de Hamming entre deux intervalles [a1 a2] et [b1 b2] vaut
(|a1-b1| + |a2-b2|) / 2
. L'addition (+) des deux intervalles [a1 a2] et [b1 b2] vaut
[a1+b1-(a1.b1) a2+b2-(a2.b2)]
. Le produit des deux intervalles [a1 a2] et [b1 b2] vaut [a1.b1 a2.b2]
. Le minimum des deux intervalles [a1 a2] et [b1 b2] vaut
[Min(a1,b1) Min(a2,b2)]
. Le maximum des deux intervalles [a1 a2] et [b1 b2] vaut
[Max(a1,b1) Max(a2,b2)]

CHAPITRE VIII : CONCLUSIONS ET PERSPECTIVES

Nous sommes arrivés au terme de ce travail et nous sommes donc en mesure de tirer des conclusions sur celui-ci et de présenter quelques prolongements possibles à ce mémoire.

8.1. CONCLUSIONS

L'objectif de notre travail est d'évaluer un logiciel de tenue des comptes pour personnes handicapées mentales.

L'évaluation d'un logiciel adressé à ce type d'utilisateur est un domaine très particulier à propos duquel nous n'avons pu trouver aucun travail.

Avant de réaliser l'évaluation en elle-même, nous avons donc été obligés de définir de manière précise ce que nous entendions par l'évaluation d'un logiciel. Nous avons, de cette manière, expliqué les buts poursuivis par notre travail.

Pour pouvoir effectivement réaliser l'évaluation, nous avons partagé quelques temps la vie de certains utilisateurs du logiciel. De cette manière, nous avons pu cerner les particularités et les difficultés liées à l'évaluation d'un logiciel adressé à cette population.

Enfin, nous avons dû composer une méthode de travail "pratique" qui nous a servi de fil conducteur pour l'évaluation.

Nous avons donc commencé par définir les objectifs sur lesquels portait notre évaluation. Ceux-ci sont en grande partie calqués sur les buts qui ont été assignés au logiciel. Nous voulions, en effet, déterminer d'une part si le logiciel Comptes était vraiment un outil d'autonomie, s'il apportait réellement des avantages par rapport à d'autres méthodes de tenue des comptes, si les utilisateurs éprouvaient un réel plaisir à l'utiliser et d'autre part, si le programme pouvait être considéré comme un outil d'apprentissage. Par ailleurs, nous étions également intéressés par l'évaluation d'autres éléments tels que l'adéquation des prérequis, la durée d'apprentissage, les qualités ergonomiques du logiciel.

Afin de se donner les moyens de répondre à ces objectifs, nous avons défini les conditions dans lesquelles l'expérimentation devrait prendre place. Nous avons également relevé un certain nombre d'outils d'appréciation qui étaient à notre disposition. Nous avons montré dans quelle mesure chaque outil était susceptible de répondre aux objectifs de l'évaluation.

Nous avons établi une grille de positionnement des différents outils disponibles qui nous a permis de faire une distinction entre les instruments d'appréciation de type non-formel et de type formel. Pour ces derniers, nous avons présenté quelques méthodes mathématiques de traitement de données.

Nous avons ensuite lancé l'expérimentation dans différentes institutions suivant les modalités définies et au moyen des outils d'appréciation utilisables à ce moment.

En dépouillant les données que nous avons pu récolter, nous avons pu répondre à chaque objectif de l'évaluation.

De plus, l'expérimentation nous a permis de critiquer les outils d'appréciation eux-mêmes. Nous avons ainsi pu dépister les lacunes et faiblesses liées à l'utilisation de chacun d'entre eux et donc montrer la nécessité de les combiner. Ils peuvent et doivent être utilisés de manière complémentaire.

Si l'évaluation nous a montré que le logiciel Comptes constituait bien un outil d'autonomie pour certains utilisateurs et qu'il permettait en outre des apprentissages, elle a également mis en évidence la nécessité d'apporter des modifications et des extensions au programme. Nous avons réalisé un bon nombre de celles-ci et nous avons élaboré une seconde version du logiciel.

A partir des critiques sur les outils d'évaluation, nous avons pu améliorer la qualité d'un de ceux-ci à savoir les traces d'utilisation et concevoir un nouvel instrument: la mesure des temps. Pour améliorer la rapidité d'exploitation de ces deux outils formels, nous avons créé un logiciel d'aide à l'évaluation. Ce programme, s'il s'est révélé utile pour l'évaluateur, semble également intéresser un bon nombre d'éducateurs.

Pour terminer cette deuxième boucle dans le cycle de vie du projet, nous avons tout naturellement commencé l'évaluation de la seconde version du logiciel sur base de la même démarche que celle utilisée pour évaluer la première version. Ceci nous a permis d'illustrer le rôle et l'intérêt des deux outils précités.

Suite aux difficultés rencontrées lors des expérimentations menées avec les deux versions du logiciel, nous avons lancé une réflexion sur une alternative possible à ces conditions. On peut résumer ces conditions de cette manière: on constitue une équipe de "juges" qui a pour mission d'apporter des éléments de réponse aux objectifs de l'évaluation; tous les membres de cette équipe rencontreront ensemble tous les utilisateurs choisis pour l'expérimentation, assisteront à quelques séances d'utilisation du logiciel et porteront chacun un jugement autonome sur l'interaction entre l'utilisateur et le logiciel. Une fois ces conditions mises en place, des outils d'appréciation plus fins et plus précis que ceux précités pourraient être utilisés. Nous avons exposé l'emploi de certains concepts des mathématiques floues comme méthode de traitement des résultats fournis par les instruments d'appréciation.

8.2. PERSPECTIVES

Nous présentons quelques suites qui pourraient être données à notre travail. Nous verrons d'abord les réalisations qui pourraient être entreprises du point de vue de la programmation du projet de gestion de budget et ensuite en matière d'évaluation.

En ce qui concerne le logiciel de tenue des comptes, un certain nombre de modifications peuvent encore être entreprises sur base des résultats des évaluations portant sur les première et seconde versions du logiciel.

Rappelons cependant que le programme Comptes n'est qu'un sous-système du projet de gestion de budget et qu'il existe une réelle demande de la part des institutions, des éducateurs et des utilisateurs eux-mêmes pour disposer d'un logiciel de gestion de budget.

Bibliographie

- Amiga DOS 1.3, Manuel d'utilisation, Français, Commodore.

- Bertier P. et Bouroche J.M., Analyse des donnnées multidimensionnelles, P.U.F. Paris, 1975.

- Briard François, Lefebvre Philippe, Meurisse Renaud et Minet Joël, Travail de psychologie, Analyse d'un logiciel budgétaire pour utilisateurs handicapés : le programme "Comptes", F.N.D.P., Namur, 1989-1990.

- Delville Jacqueline et Mercier Michel, Quand l'ordinateur est un outil d'adaptation à la vie quotidienne, Journal de réflexion sur l'informatique, n° 10, F.N.D.P., Institut d'informatique, Juin 1988.

- Déplechin Marc et Strappazzon Gianni, Un logiciel de gestion des comptes pour personnes handicapées mentales : développement et évaluation, Mémoire présenté en vue de l'obtention du titre de Licencié et Maître en informatique, F.N.D.P., Septembre 1989.

- Desmet Huguette et Pourtois Jean-Pierre,Epistémologie et instrumentation en sciences humaines, Pierre Mardaga, Editeur, 1988.

- Dis-moi qui tu aides ..., Projet interdisciplinaire science théologie, Presses universitaires de Namur, 1986.

- Gleaves Richard, Modula-2 for Pascal Programmers, Springer Verlag, 1984.

- Kaufmann Arnold, Introduction à la théorie des sous-ensembles flous à l'usage des ingénieurs (Fuzzy sets theory), Tome 1, Eléments théoriques de base, Masson et C^{ie}, Editeurs, 1973.

- Kaufmann Arnold, Introduction à la théorie des sous-ensembles flous à l'usage des ingénieurs (Fuzzy sets theory), Tome 2, Application à la linguistique, à la logique et à la sémantique, 1975.

- Schneiderman Ben, Designing the User Interface: Strategies for Effective Human Computer Interaction, Addison-Wesley Publ. Comp., 1987.

- Siegel Sidney, Non parametric statistics for the behavioral sciences, International student edition, 1988.

Réalisation, impression et reliures : SPRL Centre de Reprographie - Tél. : 081/22.87.72

Année académique 1989 - 1990

Evaluation et amélioration d'un logiciel de gestion des comptes pour personnes handicapées mentales.

ANNEXES

France CHERON **Olivier BERGER**

Promoteur : Madame Monique Noirhomme

Co-promoteur : Monsieur Michel Mercier, professeur au Département de psychologie, F.N.D.P.

Mémoire présenté en vue de l'obtention du diplôme de licence et maîtrise en informatique.

Annexes

Annexe 1: Questionnaire d'évaluation de la première version du logiciel Comptes

Annexe 2: Partie ergonomique du questionnaire d'évaluation de la seconde version du logiciel Comptes

Annexe 3: Questionnaire sur le questionnaire

Annexe 4: Mode d'emploi "technique" du logiciel Comptes

Annexe 5: Table de la distribution binomiale

Annexe 6: Présentation des fréquences et des intervalles de confiance calculés à partir des réponses au questionnaire

Annexe 7: Table de la distribution t de Student

Annexe 8: Présentation des résultats obtenus grâce à l'analyse en composantes principales

Annexe 9: Listing de la seconde version du logiciel Comptes

- Le programme Comptes
- Le programme Paramètres
- Le programme d'aide à l'évaluation

Annexe 1

Questionnaire d'évaluation de la première version du logiciel Comptes

C. Savoir-être (Champ socio-affectif)

Affectivité, créativité, autonomie et communication.

D. Un simple délassement :

II. FICHE SIGNALÉTIQUE DE L'UTILISATEUR

- Nom : .....
- Prénom : .....
- Sexe : .....
- Age : .....

- Institution : .....

- statut : travailleur 0
  non-travailleur 0

-Handicap : mental léger 0
mental modéré 0
mental sévère 0
-Handicap sensori-moteur-associé 0
si oui, précisez : .....
.....
-Q.I. et/ou âge mental et/ou niveau de réalisation .....
.....

- Somme d'argent dont dispose la personne par semaine : ..... F.

- Nombre d'occasions qu'elle a, chaque semaine, pour dépenser son argent : .....

A remplir avant l'expérimentation

Acquis en lecture	
P1 - lit son nom	0
P2 - lit des mots	0
P3 - lit des phrases courtes	0
P4 - lit un texte simple	0
P5 - lit les chiffres	0
P6 - lit les nombres	0
Acquis en écriture	
P7 - écrit son nom	0
P8 - écrit des mots	0
P9 - écrit des phrases courtes	0
P10 - écrit un texte simple	0
P11 - écrit les chiffres	0
Acquis en calcul	
P12 - établit une différence entre "un" et "beaucoup"	0
P13 - a la notion de quantité ("plus", "moins","egal")	0
P14 - dénombre	0
P15 - compte	0
P16 - additionne	0
P17 - soustrait	0
P18 - multiplie	0
P19 - divise	0
Notion de temps	
P20 - connaît les moments de la journée	0
P21 - connaît les jours de la semaine	0
P22 - connaît les jours du mois	0

A remplir avant et après l'expérimentation

	Avant	Après
Notion de l'argent		
P23 - reconnaît les pièces		
- facilement (1)	0	0
- difficilement (1/2)	0	0
- pas du tout (0)	0	0
P24 - reconnaît les billets		
- facilement (1)	0	0
- difficilement (1/2)	0	0
- pas du tout (0)	0	0
P25 - connaît la valeur relative des pièces et des billets (ex. 1 billet de 100F = 5 pièces de 20F).	0	0
P26 - est capable de vérifier si la monnaie est exacte - facilement (1)	0	0
- difficilement (1/2)	0	0
- pas du tout (0)	0	0
- déchiffre un ticket de caisse:		
P27 - peut lire les montants d'argent	0	0
P28 - peut distinguer - les différentes rubriques	0	0
P29 - le total	0	0
P30 - la somme donnée	0	0
P31 - la somme rendue	0	0
P32 - comprend le sens de "recette"		
- parfaitement (1)	0	0
- à moitié (1/2)	0	0
- pas du tout (0)	0	0
P33 - comprend le sens de "dépense"		
- parfaitement (1)	0	0
- à moitié (1/2)	0	0
- pas du tout (0)	0	0

	Avant	Après
Notion du prix des choses		
P34 - uniquement pour des objets qu'il achète - précise (1)	0	0
- vague (1/2)	0	0
- aucune (0)	0	0
P35 - également pour d'autres objets - précise (1)	0	0
- vague (1/2)	0	0
- aucune (0)	0	0
Notion de l'état de son porte-feuille		
P36 - reste tout, pas tout	0	0
P37 - reste la moitié, tout	0	0
P38 - reste peu (1/4), la moitié, tout	0	0
P39 - reste beaucoup (3/4), la moitié, peu (1/4), tout	0	0
Attention portée à l'argent		
- paie et vérifie effectivement la monnaie rendue sur - 100 FB (P40)	0	0
- 500 FB (P41)	0	0
- 1000 FB (P42)	0	0
- essaye de mémoriser les dépenses - d'une demi-journée (P43)	0	0
- de plusieurs jours (P44)	0	0
- d'une semaine (P45)	0	0
P46 - utilise un support papier-crayon pour mémoriser ses dépenses	0	0
P47 - conserve son ticket de caisse	0	0

A remplir avant l'expérimentation

Familiarité avec l'ordinateur	
P48 - a déjà eu accès à un ordinateur	0
Si oui,	
P49 - contexte : - divertissement	0
P50 - éducationnel	0
- nombre de mois	_____
- fréquence des séances	
(par mois, par semaine)	____/____
- durée des séances	_____ min.
- sait mettre l'ordinateur en marche	
P51 - avec aide	0
P52 - sans aide	0
- sait utiliser la souris pour	
P53 - la positionner : - facilement (1)	0
- difficilement (1/2)	0
- pas du tout (0)	0
P54 - cliquer : - facilement (1)	0
- difficilement (1/2)	0
- pas du tout (0)	0
- sait utiliser un clavier pour	
P55 - taper son nom	0
P56 - taper un chiffre	0
P57 - taper du texte	0

A compléter pendant l'expérimentation toutes les 3 séances

	Séance - date de la séance - durée de la séance						
	Saisie du nom (cfr. dessin des écrans : page1)						
E1	- La personne sait écrire son nom						
	- oui, sans aide (1)	0	0	0	0	0	0
	- oui, avec aide (1/2)	0	0	0	0	0	0
	- non (0)	0	0	0	0	0	0
	Saisie de la somme initiale par souris(page 2)						
E2	- La personne comprend la différence entre "effacer" et "recommencer" - facilement (1)	0	0	0	0	0	0
	- difficilement (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
	Saisie du montant d'une recette (page 8) ou d'une dépense (page 13) par souris						
E3	- La personne comprend la différence entre "quitter" et "fini" - facilement (1)	0	0	0	0	0	0
	- difficilement (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E4	- L'icône "recette" ou "dépense" est utile à la personne	0	0	0	0	0	0

	Saisie d'une somme (initiale (p.2),recette(p.8), dépense (p.13)) **par souris**						
E5	- La personne reconnaît les billets	0	0	0	0	0	0
E6	- La personne reconnaît les pièces	0	0	0	0	0	0
E7	- La personne remarque qu'une trace apparaît lorsqu'elle a cliqué sur un billet ou une pièce	0	0	0	0	0	0
E8	- La personne est perturbée par le fait que la trace n'apparaît plus lorsqu'elle entre plus de pièces ou de billets que ne le permet l'écran	0	0	0	0	0	0
E9	- Il arrive que la personne clique sur un billet ou une pièce se trouvant sur la partie droite de l'écran	0	0	0	0	0	0
E10	- Il arrive que la personne clique sur un autre billet (pièce) que celui désiré par imprécision	0	0	0	0	0	0
E11	- La personne comprend le sigle "oreille"	0	0	0	0	0	0
E12	- La personne se sert parfois du sigle "oreille"	0	0	0	0	0	0
E13	- Le terme "fini" est bien compris - oui, sans aide (1)	0	0	0	0	0	0
	- oui, avec aide (1/2)	0	0	0	0	0	0
	- non (0)	0	0	0	0	0	0
E14	- Le total est utile à la personne	0	0	0	0	0	0
	Entrée d'une somme (initiale (p.3), recette (p.9), dépense (p.14)) **par clavier**						
E15	- La personne peut taper un chiffre au clavier						
	- facilement (1)	0	0	0	0	0	0
	- difficilement (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E16	- Il arrive que la personne entre des caractères qui ne sont pas des chiffres	0	0	0	0	0	0
E17	- La personne est perturbée par le fait de devoir taper sur la touche "enter"/"return" ↵	0	0	0	0	0	0
	Menu (page 4)						
E18	- les termes utilisés sont compris par la personne						
	- parfaitement (1)	0	0	0	0	0	0
	- à moitié (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E19	- Les dessins sont compris par la personne	0	0	0	0	0	0
E20	- La personne clique sur "fin" par inadvertance	0	0	0	0	0	0

	Entrée du poste budgétaire d'une dépense (10)						
E21	- La personne met presque toutes les dépenses dans le poste "divers"	0	0	0	0	0	0

	Entrée du poste budgétaire d'une recette (P.5) **ou d'une dépense** (P.10)						
E22	- La personne se base sur les dessins pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E23	- La personne se base sur le texte pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E24	- La personne peut classer les recettes et les dépenses dans des postes comme "vêtements", "loisirs", "transports",...	0	0	0	0	0	0
	Entrée du jour d'une recette (p.6) **ou d'une dépense** (p.11) **(cycle d'une semaine)**						
E25	- Pour la réglette du temps, la personne se base sur la couleur pour percevoir les jours passés et à venir - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E26	- La personne se base sur le texte pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0

	Entrée du moment d'une recette (p.7) **ou d'une dépense** (p.12) **(cycle d'une journée)**						
E27	- La personne se base sur les dessins pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E28	- La personne se base sur le texte pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0

	Etat du porte monnaie (représentation graphique) (p.15)						
E29	- La personne comprend que la somme qu'elle voit est celle qu'elle a communiquée au programme au début de l'exercice	0	0	0	0	0	0
E30	- La personne perçoit son erreur (s'il y en a une) par le biais du thermomètre	0	0	0	0	0	0
E31	- La personne comprend qu'elle doit cliquer sur "fini" pour passer à l'écran suivant	0	0	0	0	0	0

	Etat du porte monnaie (représentation par nombre) (P.16)						
E32	- La personne comprend que la somme qu'elle voit est celle qu'elle a communiquée au programme au début de l'exercice	0	0	0	0	0	0

	Récapitulatif des recettes (P.17) **et des dépenses** (p.18)						
	- La personne comprend la signification						
E33	des colonnes : - jour	0	0	0	0	0	0
E34	- poste	0	0	0	0	0	0
E35	- montant	0	0	0	0	0	0
E36	- total	0	0	0	0	0	0
E37	- La personne comprend la ligne "solde"	0	0	0	0	0	0
E38	- La personne comprend que les différentes lignes représentent les différentes recettes et dépenses qu'elle a enregistrées	0	0	0	0	0	0
E39	- La personne perçoit le changement de la couleur du thermomètre	0	0	0	0	0	0
E40	- La personne a le temps de le voir fluctuer	0	0	0	0	0	0
E41	- La personne comprend la signification des couleurs	0	0	0	0	0	0

	Balance (p.19)						
E42	- La personne comprend le message	0	0	0	0	0	0
E43	- La personne comprend la symbolique de la balance	0	0	0	0	0	0

	Généralités (pages 1 à 19)						
E44	- Pour cette personne, les demandes de confirmation sont : - indispensables (1)	0	0	0	0	0	0
	- utiles (1/2)	0	0	0	0	0	0
	- inutiles (0)	0	0	0	0	0	0
E45	- Il arrive que la personne place mal le suiveur de la souris à cause de la forme de celui-ci	0	0	0	0	0	0
E46	- La personne clique sur des objets non-interactifs :						
	- souvent (0)	0	0	0	0	0	0
	- parfois (1/2)	0	0	0	0	0	0
	- jamais (1)	0	0	0	0	0	0
E47	- En général, la personne se base sur les couleurs pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E48	- En général, la personne se base sur le texte pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E49	- En général, la personne se base sur les dessins pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E50	- En général, la personne se base sur le son pour se retrouver - uniquement (1)	0	0	0	0	0	0
	- en partie (1/2)	0	0	0	0	0	0
	- pas du tout (0)	0	0	0	0	0	0
E51	- L'accent anglais est très gênant pour la compréhension	0	0	0	0	0	0
E52	- Une icône "oreille" serait utile dans tous les écrans	0	0	0	0	0	0
E53	- La personne est perturbée par le fait que le thermomètre ne fluctue pas lorsque son "avoir" dépasse le maximum fixé par les paramètres	0	0	0	0	0	0
E54	- La personne a le temps de voir le thermomètre fluctuer dans les différents écrans	0	0	0	0	0	0
E55	- La personne comprend les fluctuations du thermomètre	0	0	0	0	0	0
E56	- La personne comprend ce que signifie la barre jaune du thermomètre	0	0	0	0	0	0
E57	- La succession des écrans semble poser des problèmes à la personne	0	0	0	0	0	0
E58	- Le montant indiqué au dessus du thermomètre trouble la personne - fortement (0)	0	0	0	0	0	0
	- un peu (1/2)	0	0	0	0	0	0
	- pas du tout (1)	0	0	0	0	0	0

A COMPLÉTER APRES LES SÉANCES D'EXPÉRIMENTION.

I. Modifications observées après l'utilisation du logiciel.

Voir "fiche signalitétique de l'utilisateur" colonne **APRES**

II. Critique du logiciel

A) Sur le plan technique

a) Possibilités de réglage de divers paramètres :

b) Facilité d'utilisation (mise en route, menus, choix, adaptation à divers handicaps, possibilité d'annuler un ordre en cas d'erreur, etc) :

c) Qualité du graphisme (notamment dimension des caractères et des dessins), des animations, du son :

B. Sur le plan pédagogique :

a) Interactivité : clarté dans la transmission de messages.

b) Adaptabilité : Possibilité d'adaptation à divers niveaux de réalisation (âge mental - Q.I.)

c) Utilité : comparaison avec le(s) système(s) utilisés jusqu'alors (ce que ce logiciel apporte de différent) :

III. Propositions d'amélioration

IV. Variation des objectifs après utilisation.

A. Lesquels parmi vos objectifs de départ ont-ils été satisfaits (en les énumérant par ordre décroissant) ? :

B. Percevez-vous de nouveaux objectifs ? :

V. Critères d'utilisation :

A. Prérequis jugés indispensables :

a) Sur le plan affectif :

b) Sur le plan moteur :

c) Sur le plan sensoriel :

d) Sur le plan des connaissances scolaires, culturelles, professionnelles:

B. A qui faut-il déconseiller l'usage de ce logiciel ? :

VI. Avis de l'utilisateur.

- Décris l'activité réalisée (fonction perçue du logiciel)

- Ce logiciel t'aide-t-il ? En quoi ?

- (Si la personne utilisait précedemment une autre méthode pour faire ses comptes). Quelle méthode préfères-tu ? Pourquoi ?

- Est-ce que ce logiciel te donne envie d'aller plus loin c'est-à-dire par exemple d'apprendre à gérer ton budget (prévisions, économies,...)

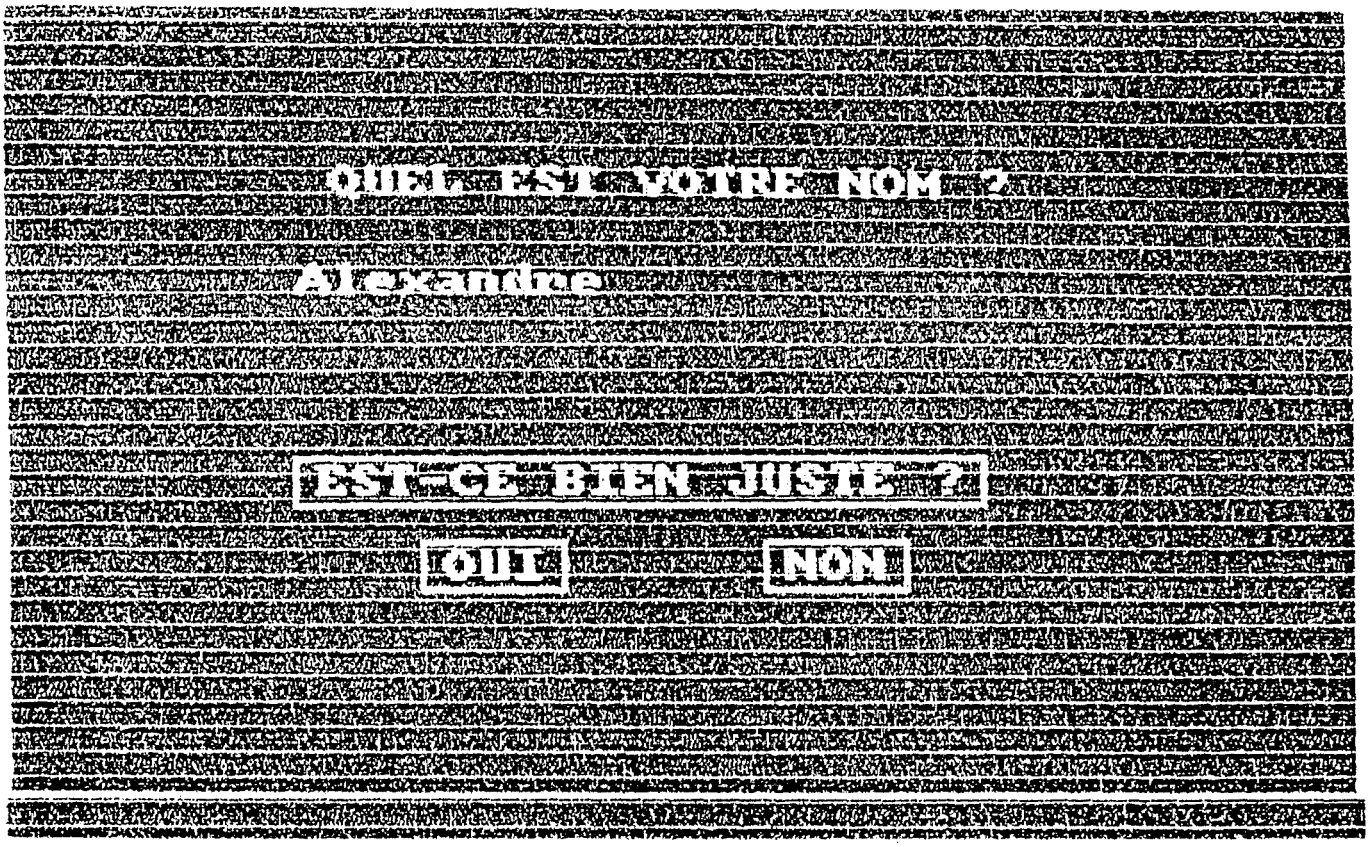
- Quelles sont les grosses difficultés que tu as rencontrées ?

- Cette activité t'amuse-t-elle ?

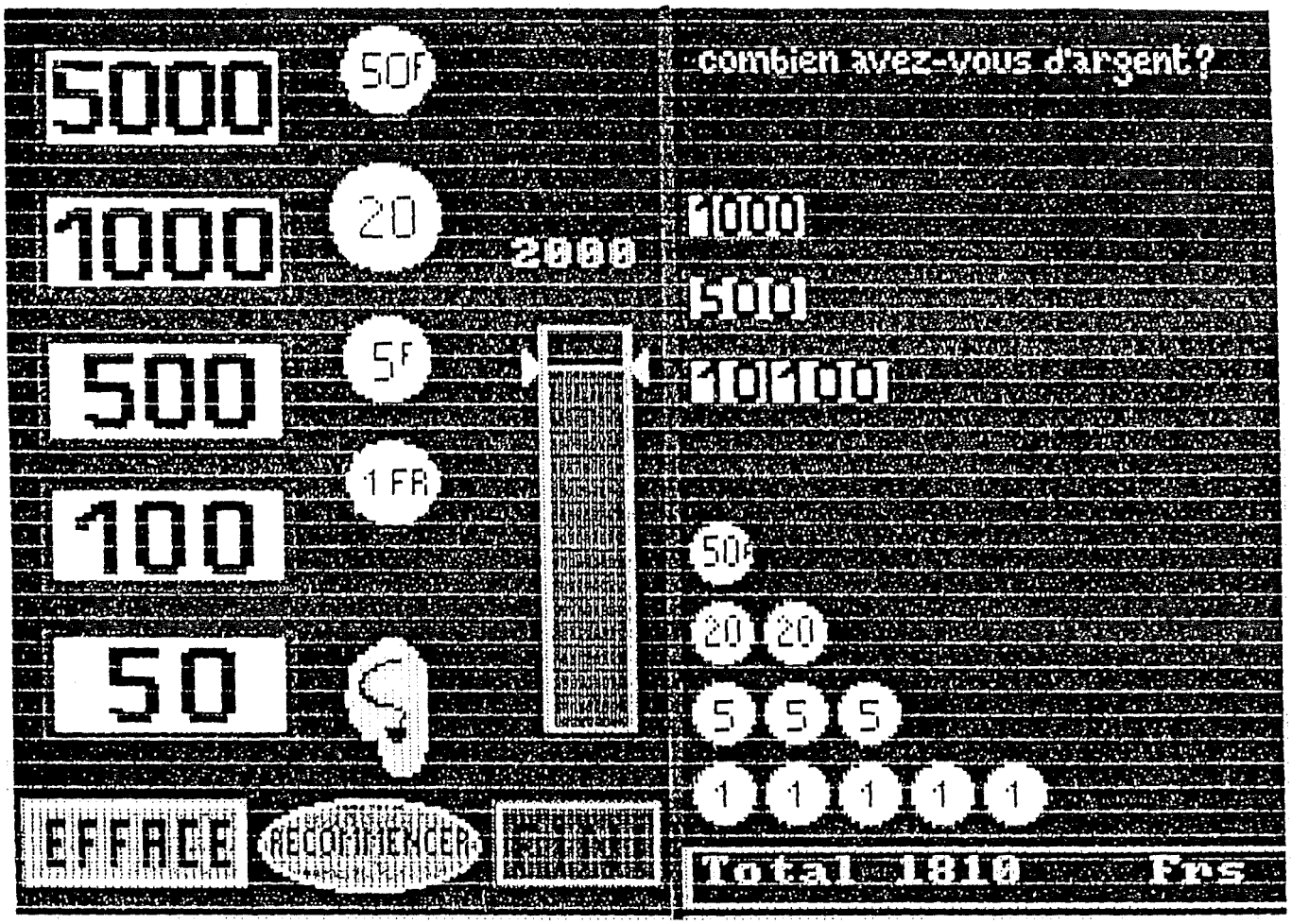
- Quelles sont, selon toi, les modifications à apporter au logiciel ?

- Aimerais-tu utiliser l'ordinateur à d'autres fins ?

  Si oui, lesquelles ?



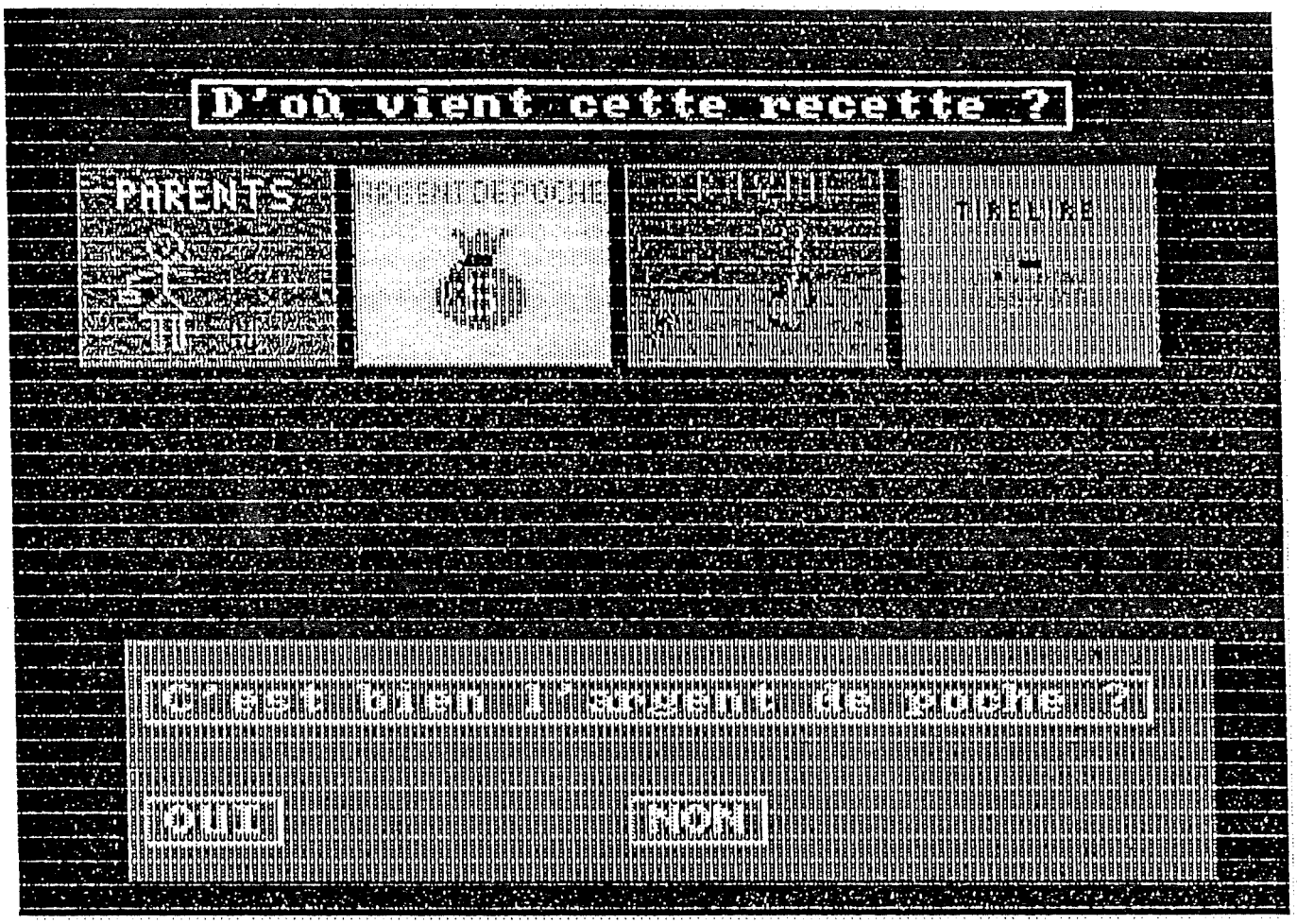
Commentaires et suggestions :



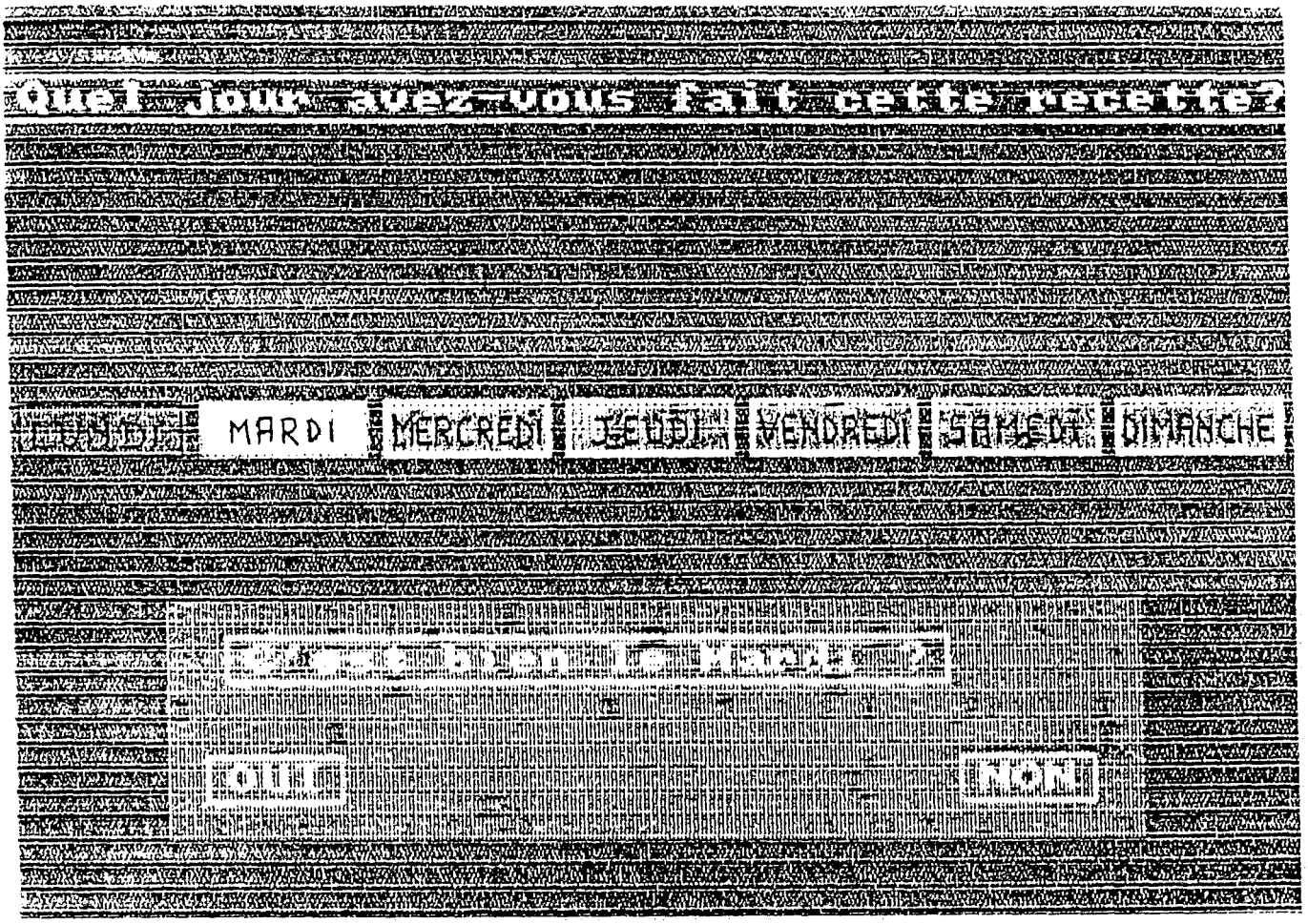
Commentaires et suggestions :

Commentaires et suggestions :

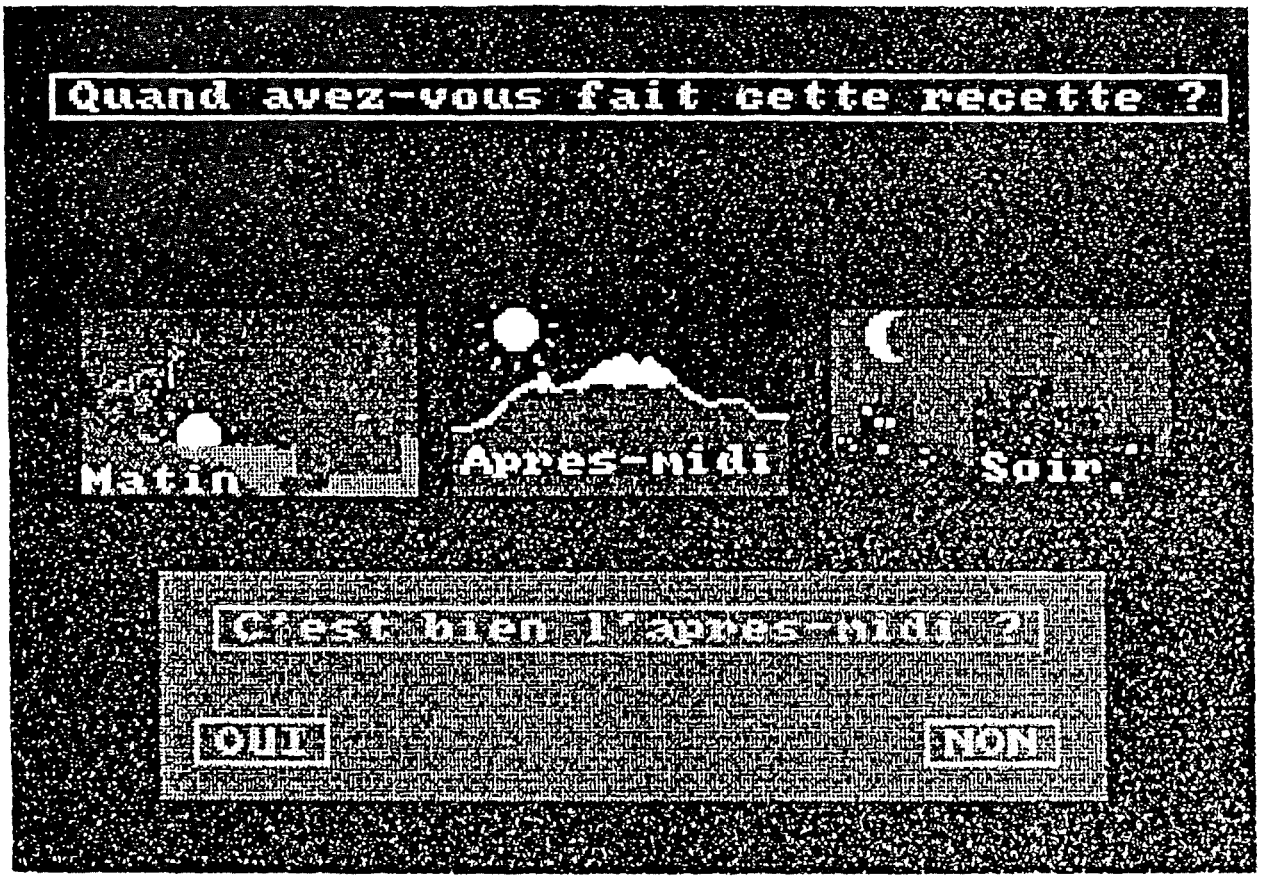
Commentaires et suggestions :



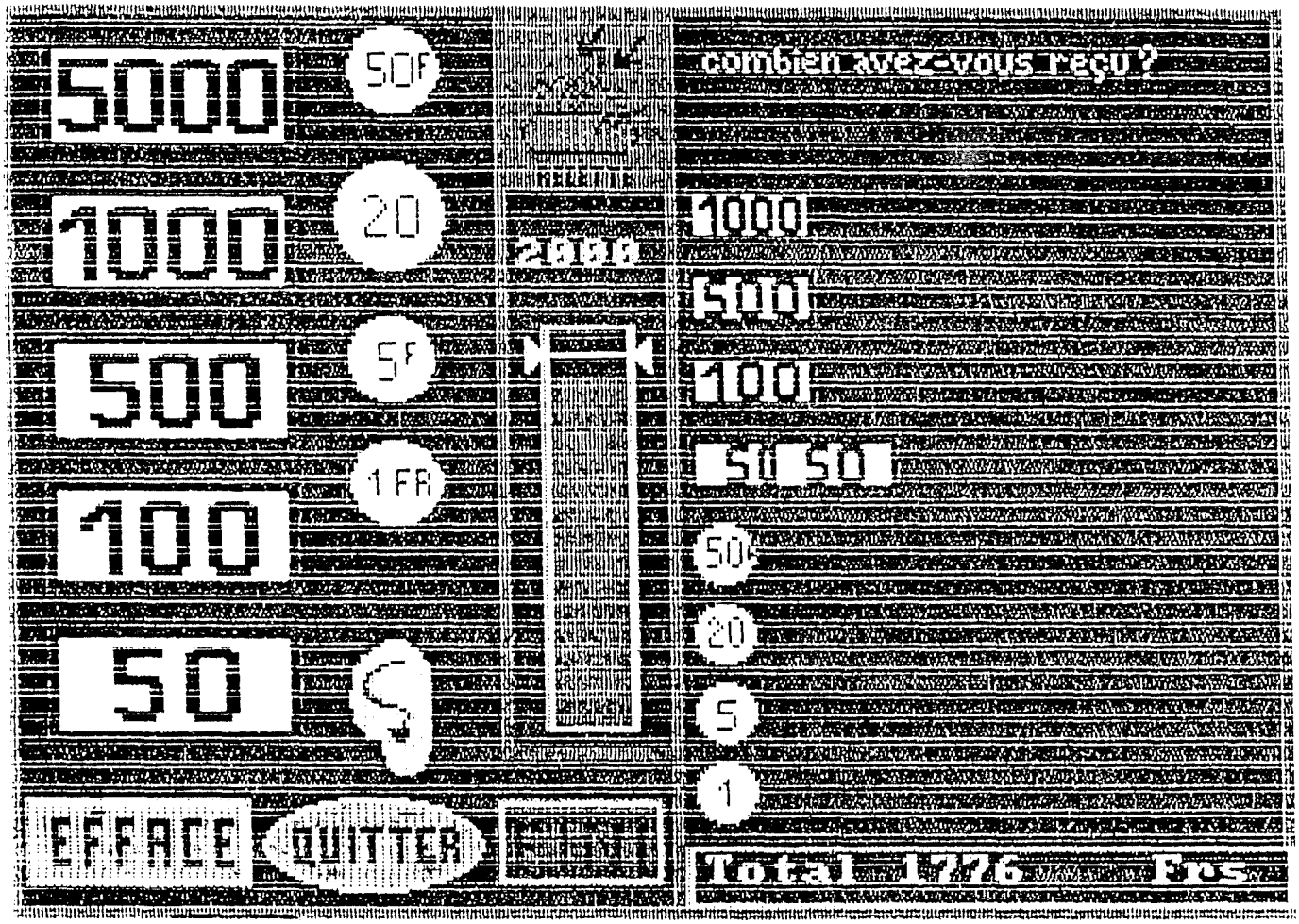
Commentaires et suggestions :



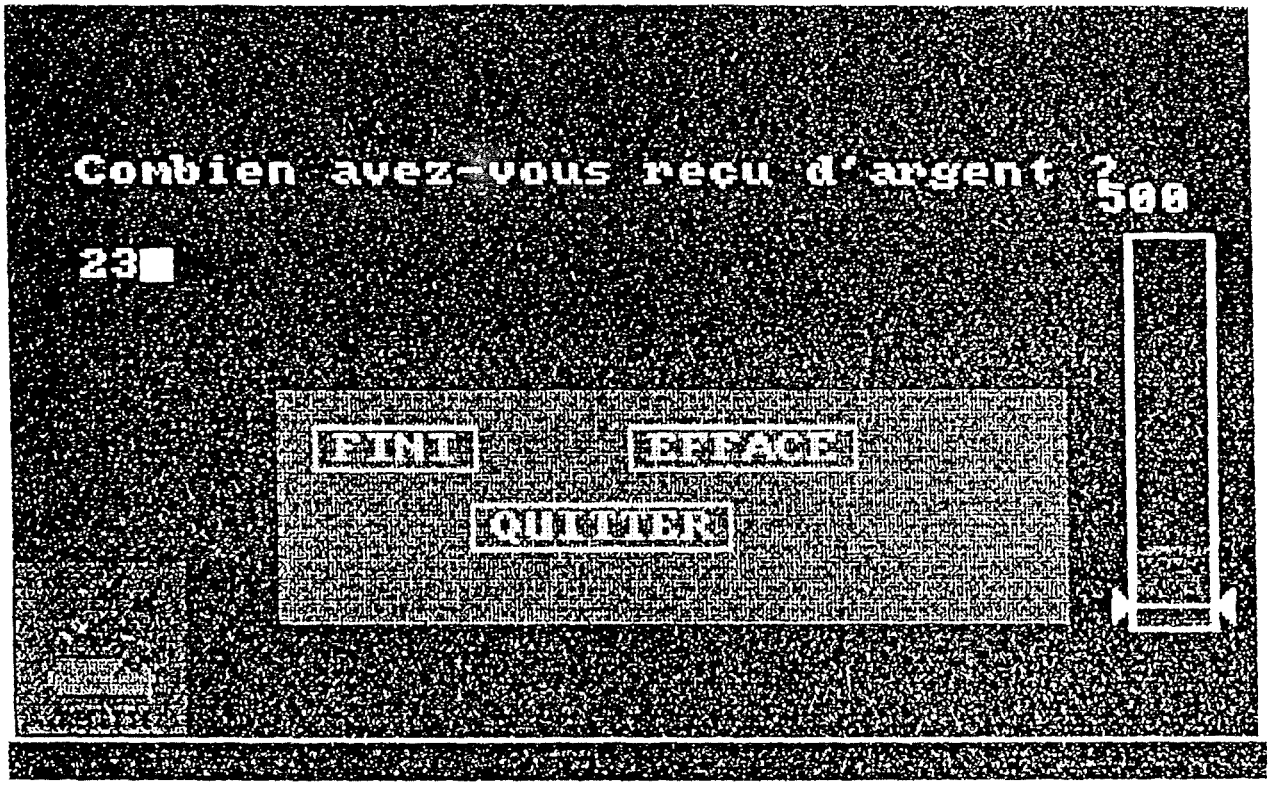
Commentaires et suggestions :



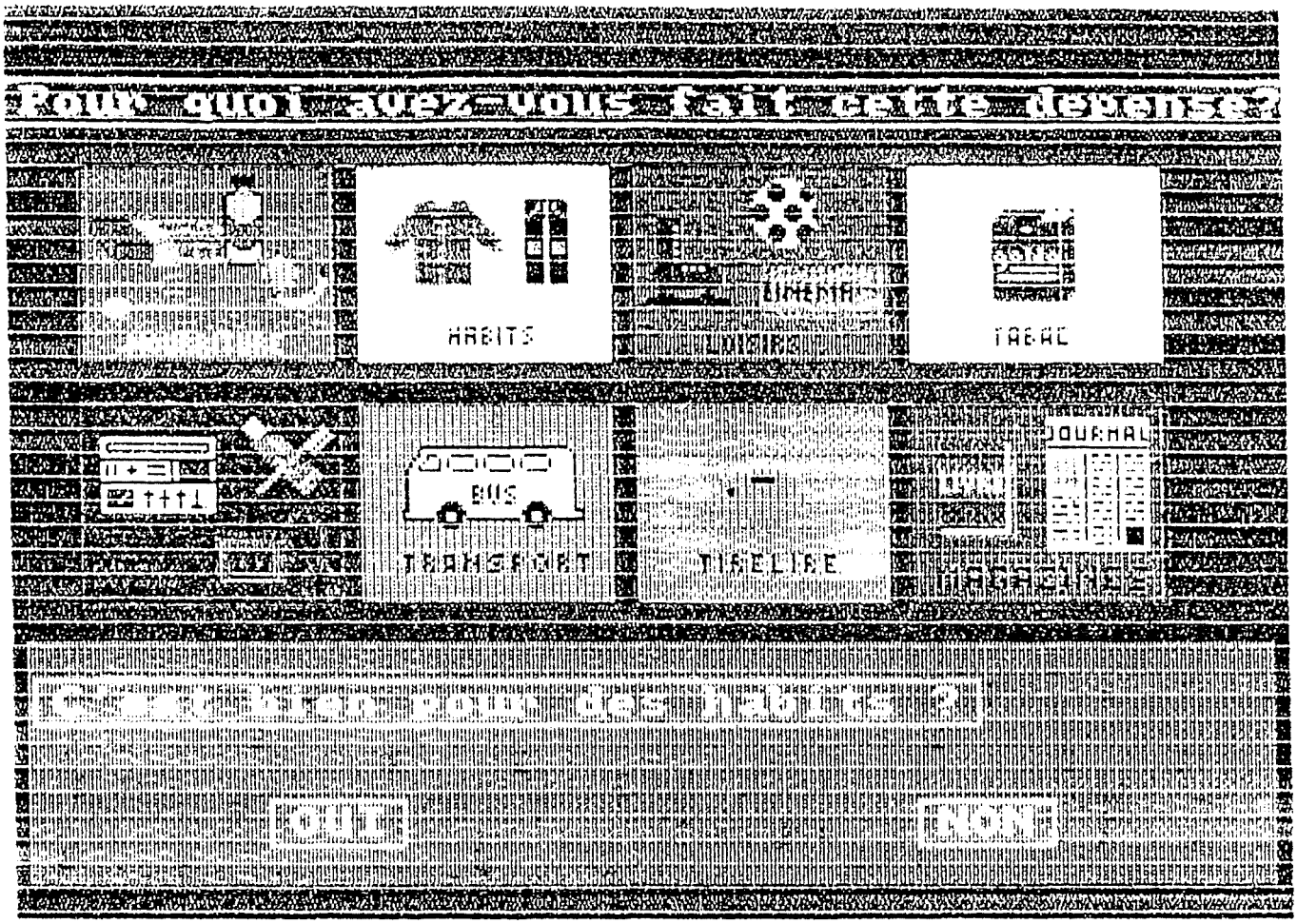
Commentaires et suggestions :



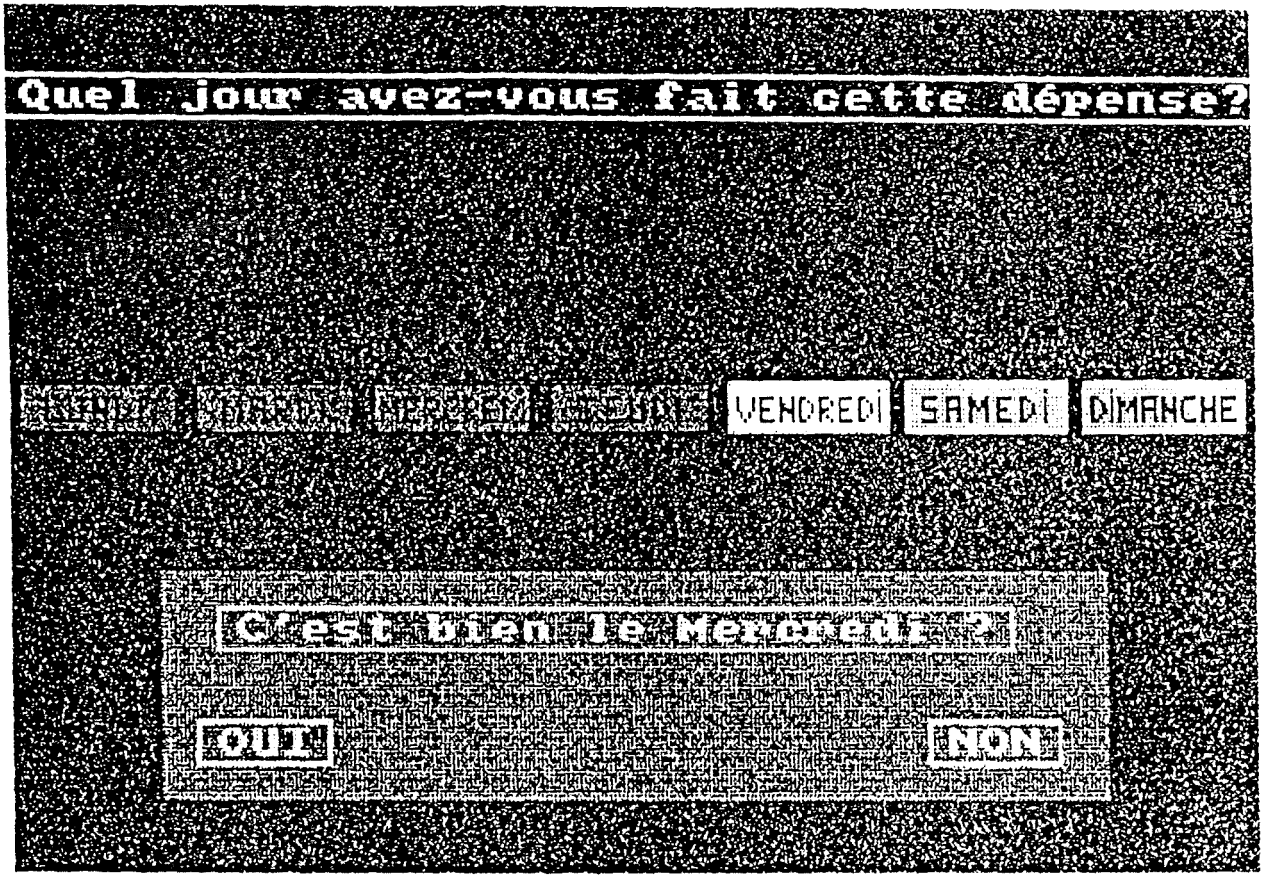
Commentaires et suggestions :



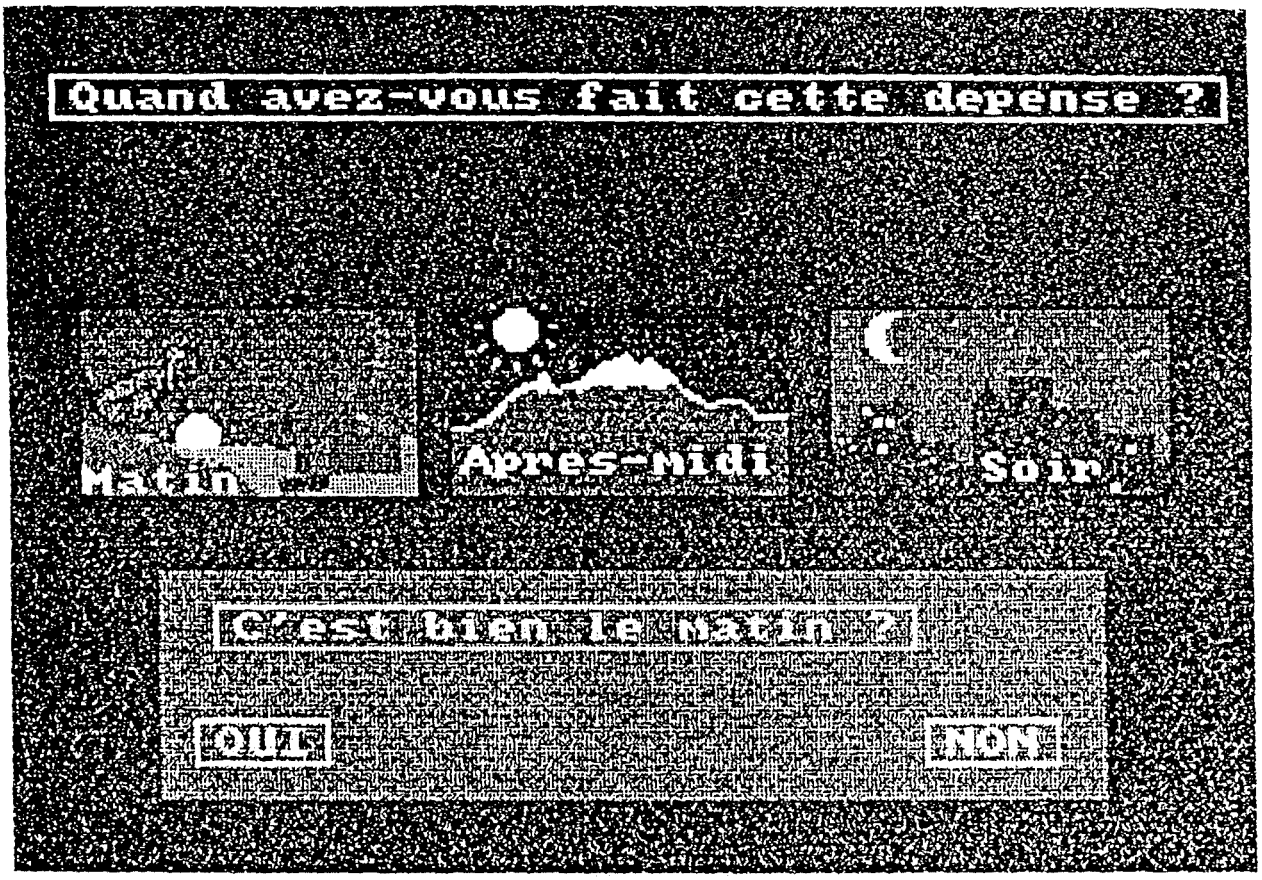
Commentaires et suggestions :



Commentaires et suggestions :

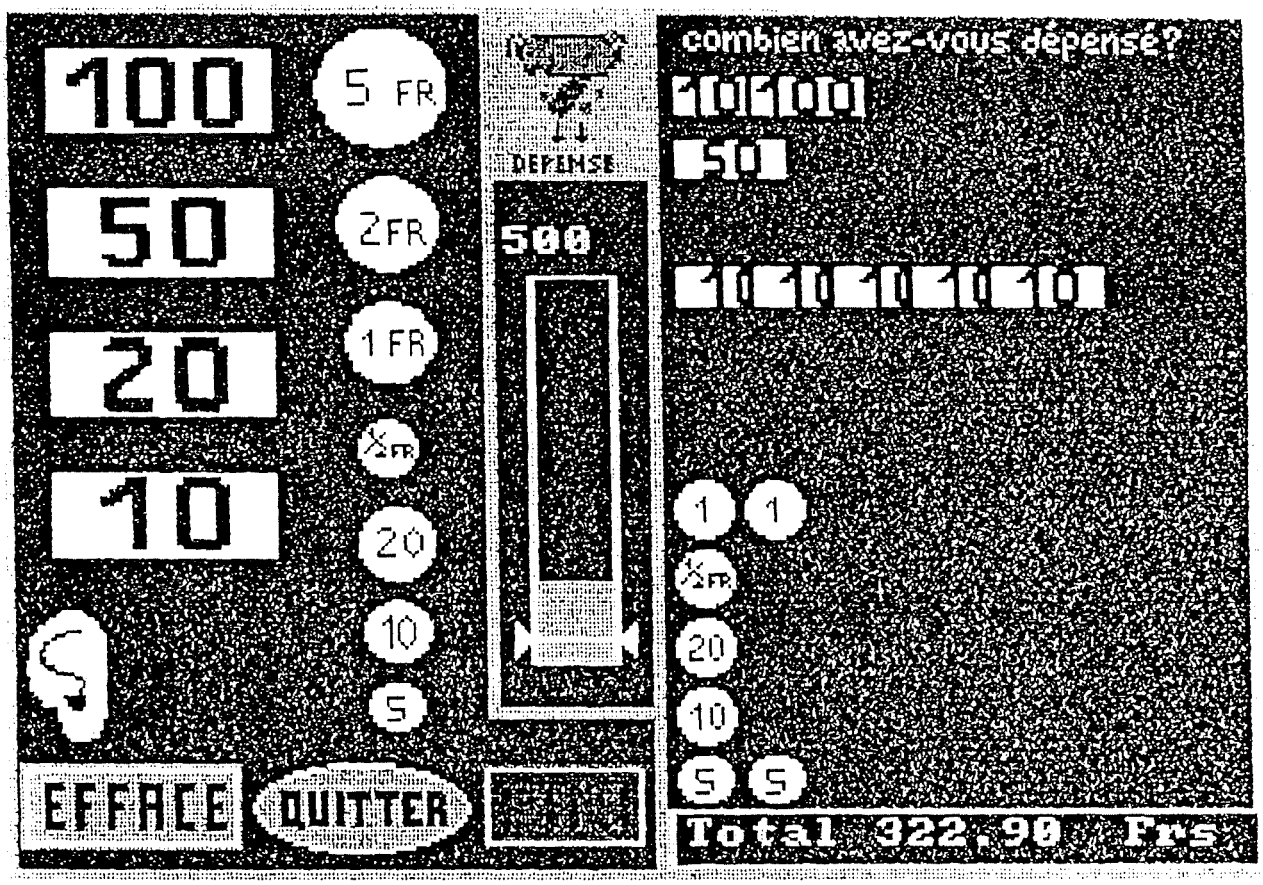


Commentaires et suggestions :

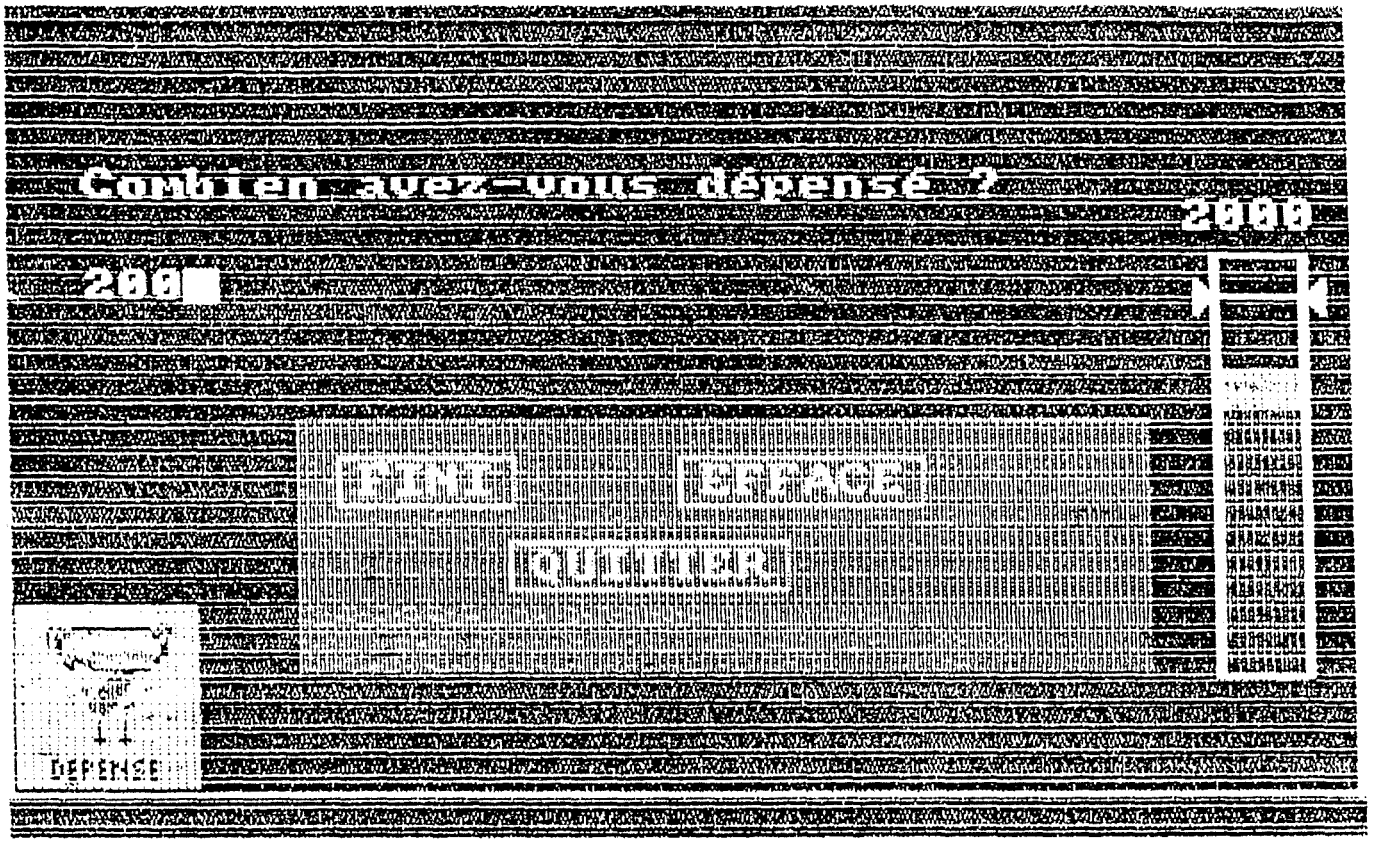


Commentaires et suggestions :

P 13

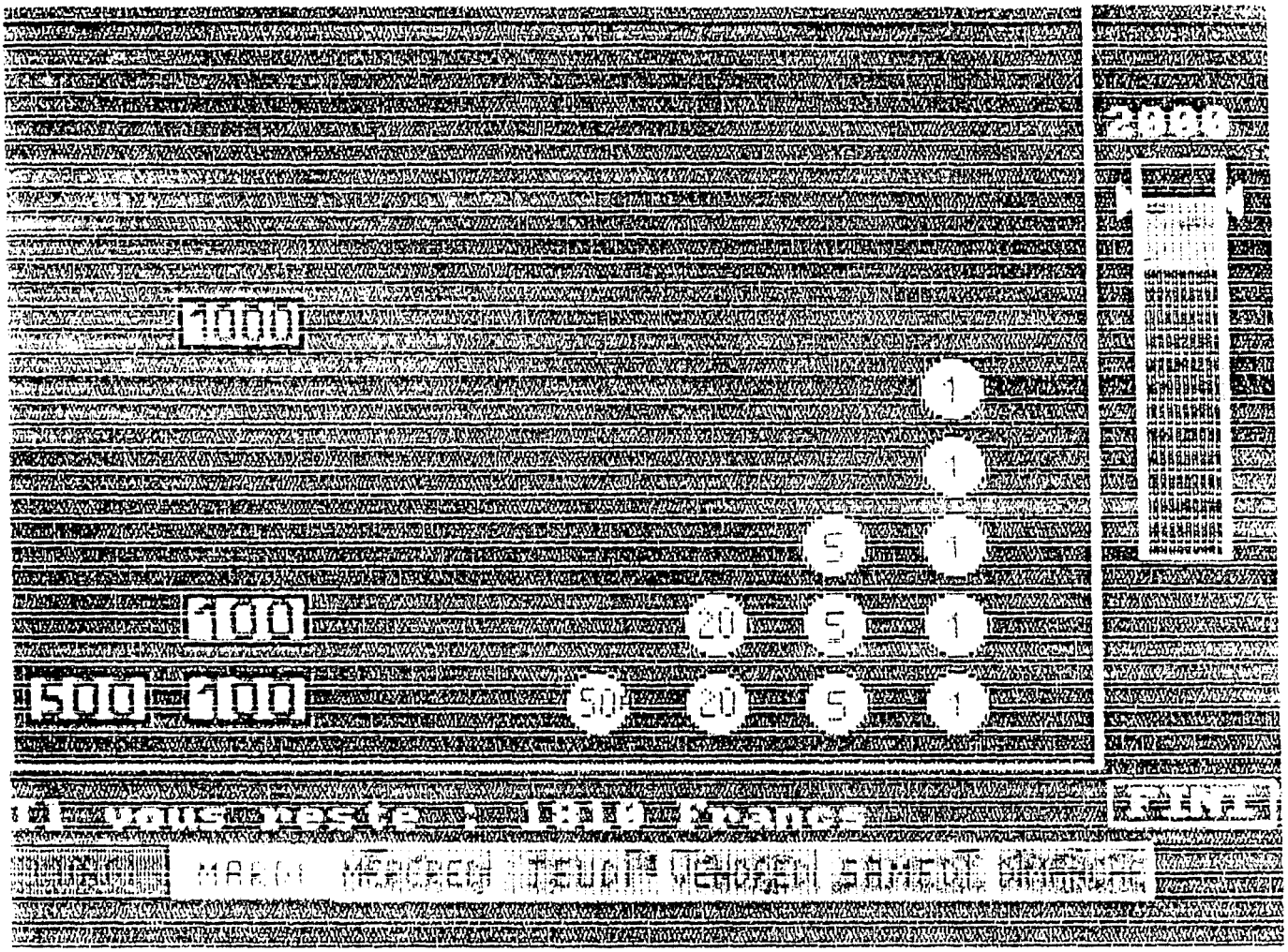


Commentaires et suggestions :

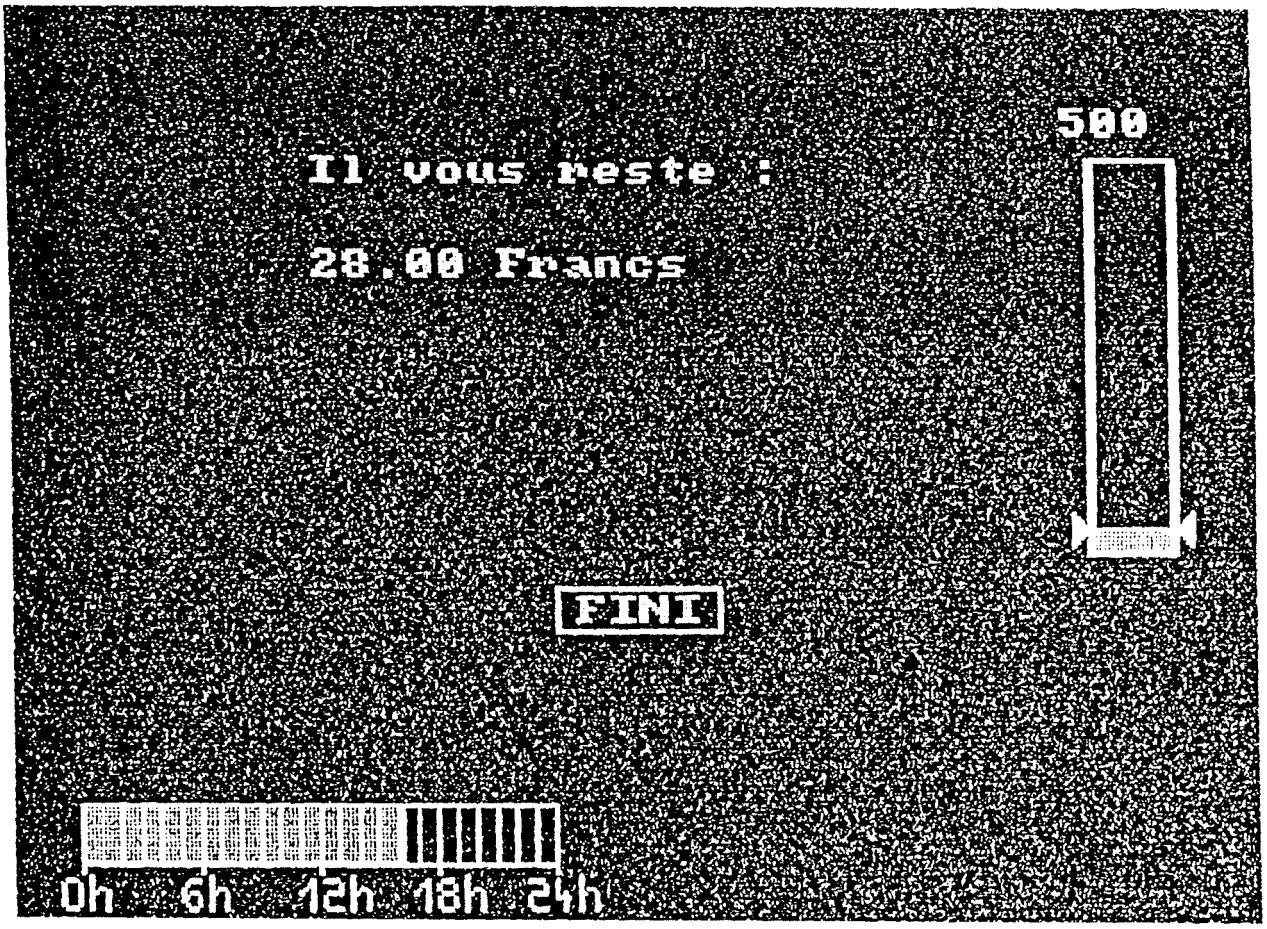


Ecran de saisie d'une dépense (par clavier)

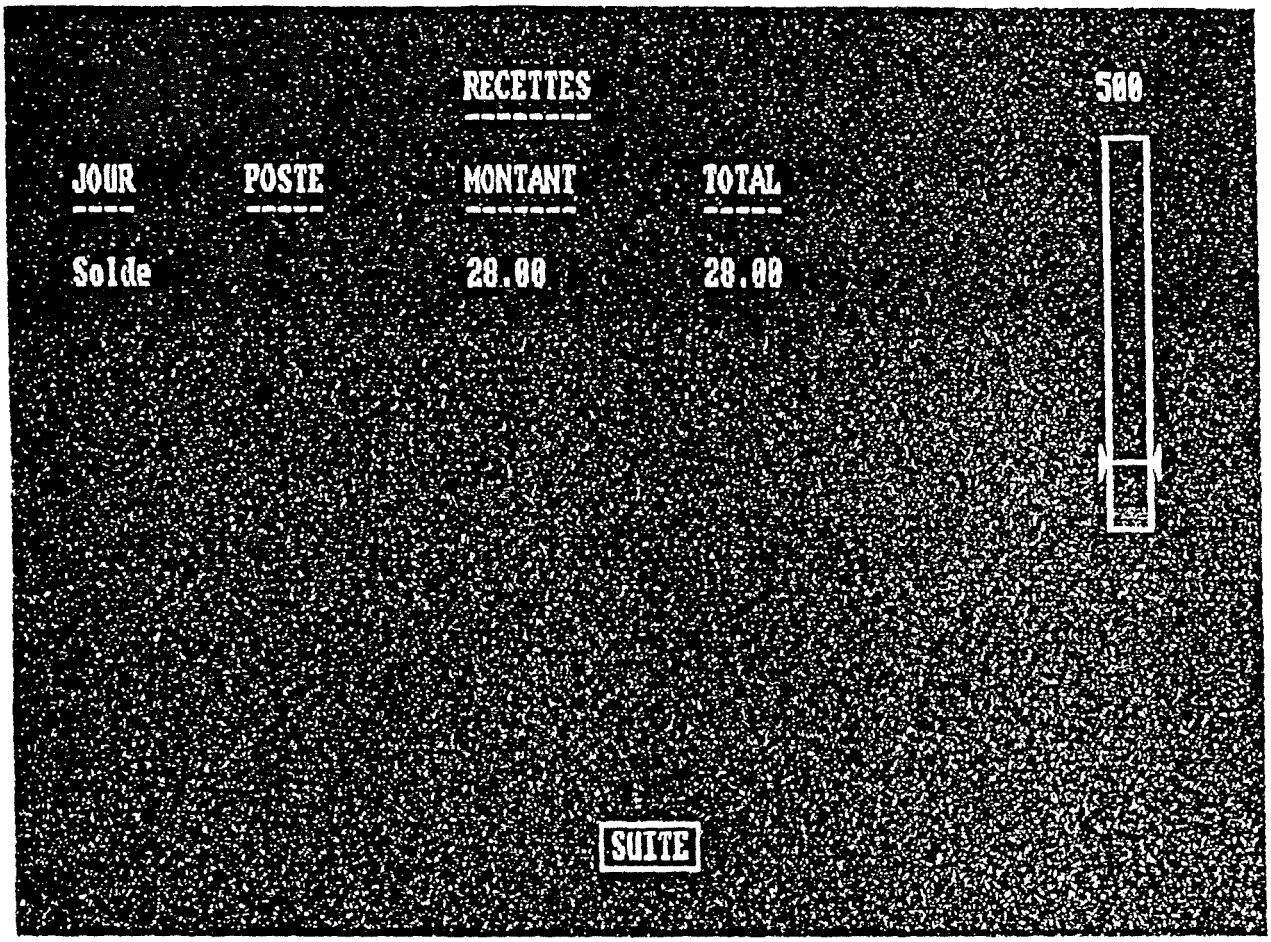
Commentaires et suggestions :



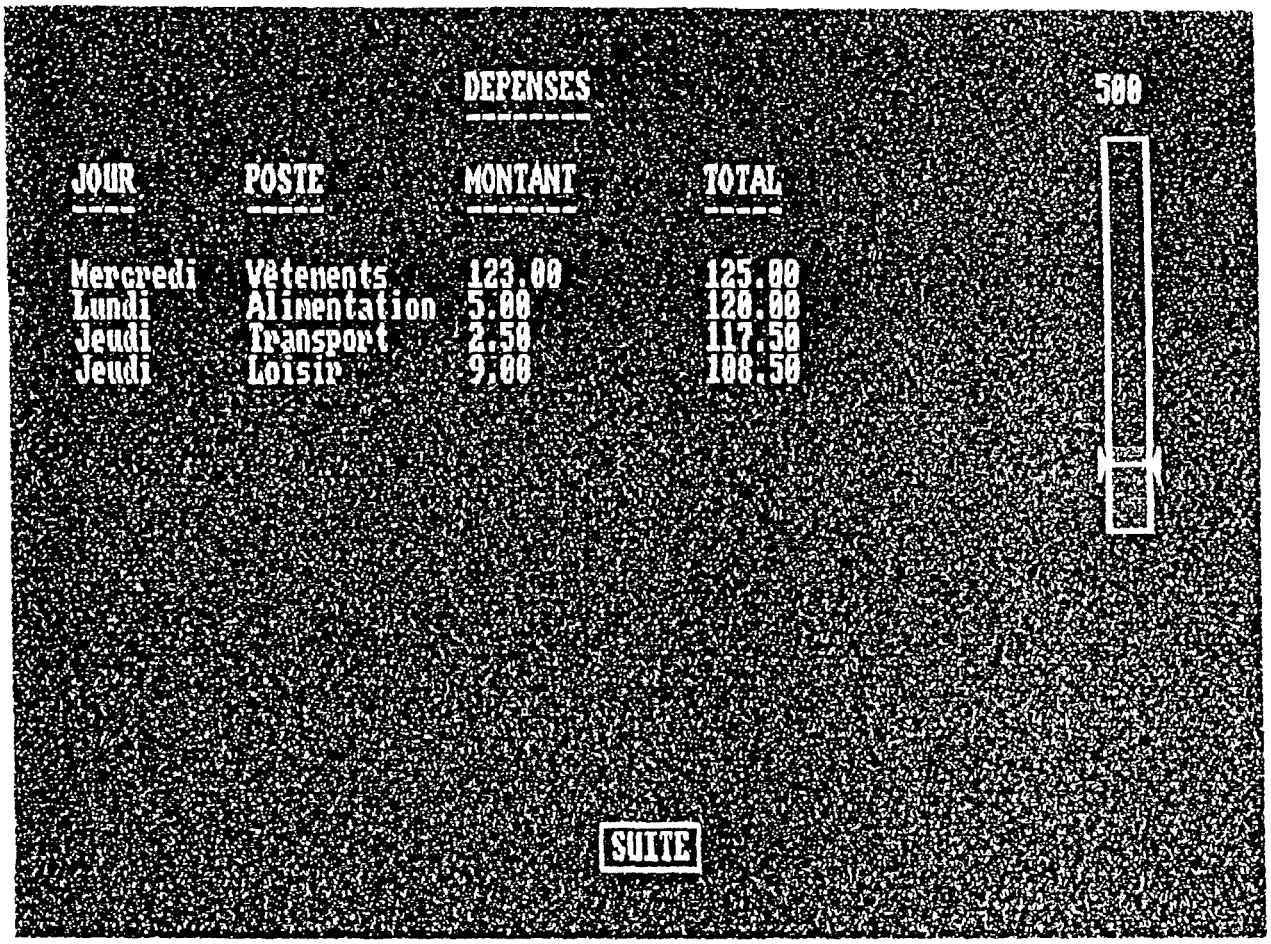
Commentaires et suggestions :



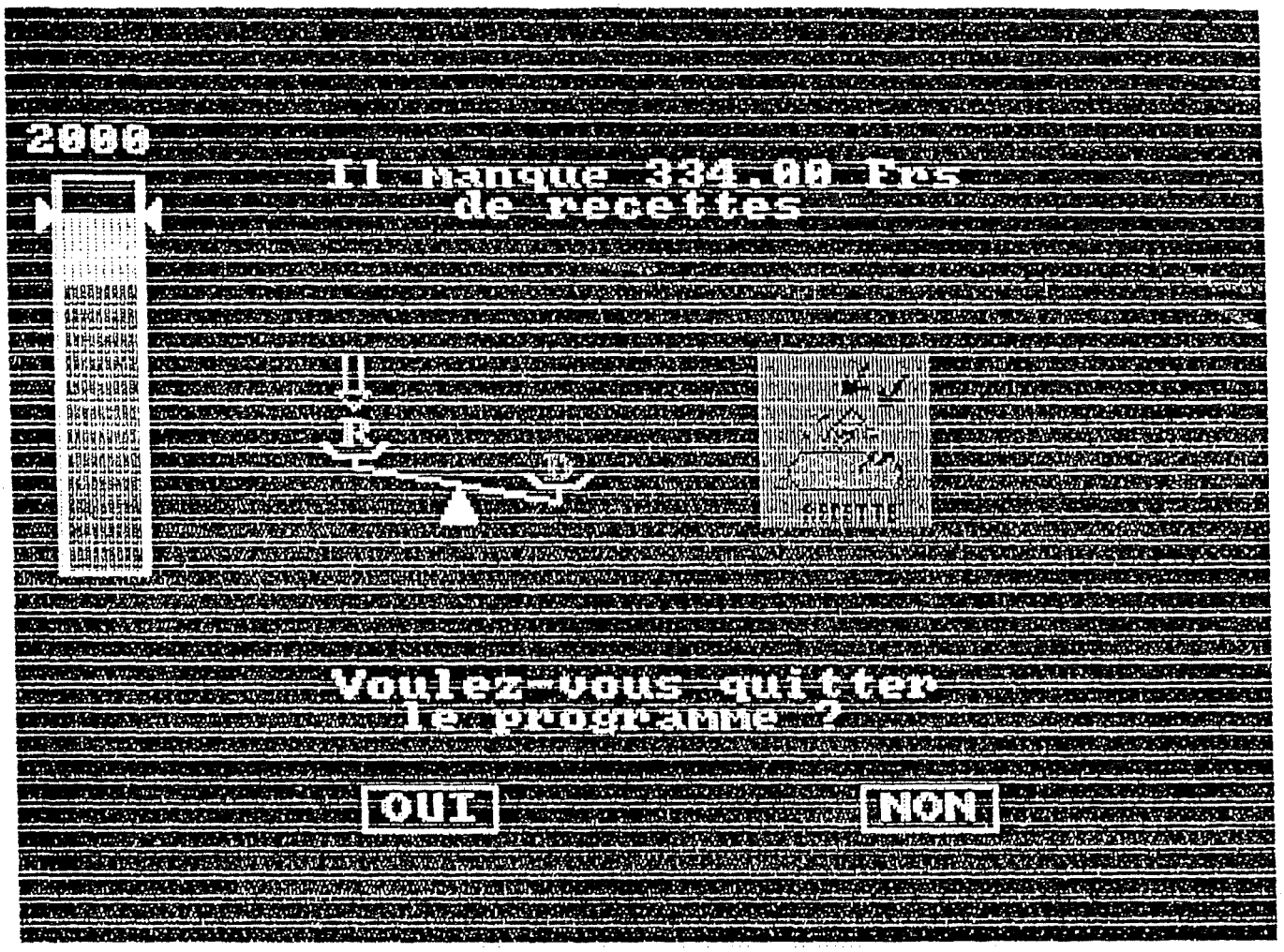
Commentaires et suggestions :



Commentaires et suggestions :



Commentaires et suggestions :



Commentaires et suggestions :

Annexe 2

Partie ergonomique du questionnaire d'évaluation de la seconde version du logiciel Comptes

A compléter pendant l'expérimentation après chaque séance

Séance - date de la séance - durée de la séance						
Saisie du nom (cfr. dessin des écrans : page 1) - La personne sait écrire son nom - oui, sans aide - oui, avec aide - non	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
Saisie du nom par souris - Pour cette personne, la possibilité d'entrer son nom par la souris est - indispensable - très intéressante - intéressante - peu intéressante - sans intérêt	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
- La personne clique spontanément sur "OK" après avoir entré son nom	0	0	0	0	0	0
- La personne comprend qu'un clavier est dessiné sur l'écran	0	0	0	0	0	0
- La personne comprend qu'elle doit cliquer sur les dessins de lettres pour entrer son nom - oui, sans aide - oui, avec aide - non	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0

Saisie de la somme initiale par souris(page 2)							
- La personne comprend la différence entre "effacer" et "recommencer"	- facilement	0	0	0	0	0	0
	- difficilement	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- L'icône "porte-monnaie" est utile à la personne		0	0	0	0	0	0
Saisie du montant d'une recette (page 8) **ou d'une dépense** (page 13) **par souris-billets**							
- La personne comprend la différence entre "quitter" et "ok"	- facilement	0	0	0	0	0	0
	- difficilement	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- L'icône "recette" ou "dépense" est utile à la personne		0	0	0	0	0	0
Saisie d'une somme (initiale (p.2), recette (p.8), dépense (p.13)) **par souris-billets**							
- La personne reconnaît les billets		0	0	0	0	0	0
- La couleur des différents billets est utile à la personne		0	0	0	0	0	0
- La personne reconnaît les pièces		0	0	0	0	0	0
- La personne remarque qu'une trace apparaît lorsqu'elle a cliqué sur un billet ou une pièce		0	0	0	0	0	0
- La personne est perturbée par le fait que la trace n'apparaît plus lorsqu'elle entre plus de pièces ou de billets que ne le permet l'écran		0	0	0	0	0	0
- Il arrive que la personne clique sur un billet ou une pièce se trouvant sur la partie droite de l'écran		0	0	0	0	0	0
- Il arrive que la personne clique sur un autre billet (pièce) que celui désiré par imprécision		0	0	0	0	0	0
- La personne comprend le sigle "oreille"		0	0	0	0	0	0
- La personne se sert parfois du sigle "oreille"		0	0	0	0	0	0
- Le terme "ok" est bien compris	- oui, sans aide	0	0	0	0	0	0
	- oui, avec aide	0	0	0	0	0	0
	- non	0	0	0	0	0	0
- Le total est utile à la personne		0	0	0	0	0	0

Entrée d'une somme (initiale (p.3), recette (p.9), dépense (p.14)) **par clavier**						
- La personne peut taper un chiffre au clavier						
- facilement	0	0	0	0	0	0
- difficilement	0	0	0	0	0	0
- pas du tout	0	0	0	0	0	0
- Il arrive que la personne entre des caractères qui ne sont pas des chiffres	0	0	0	0	0	0
- La personne est perturbée par le fait de devoir taper sur la touche "enter"	0	0	0	0	0	0
Entrée d'une somme (initiale, recette, dépense) **par souris-calculette**						
- Pour cette personne, la possibilité d'entrer les montants par la souris est - indispensable	0	0	0	0	0	0
- très intéressante	0	0	0	0	0	0
- intéressante	0	0	0	0	0	0
- peu intéressante	0	0	0	0	0	0
- sans intérêt	0	0	0	0	0	0
- La personne clique spontanément sur "OK" après avoir entré son montant	0	0	0	0	0	0
- La personne comprend qu'un pavé numérique est dessiné sur l'écran	0	0	0	0	0	0
- La personne comprend qu'elle doit cliquer sur les dessins de chiffres pour entrer son montant						
- oui, sans aide	0	0	0	0	0	0
- oui, avec aide	0	0	0	0	0	0
- non	0	0	0	0	0	0

Menu (page 4)						
- les termes utilisés sont compris par la personne						
- parfaitement	0	0	0	0	0	0
- à moitié	0	0	0	0	0	0
- pas du tout	0	0	0	0	0	0
- Les dessins sont compris par la personne	0	0	0	0	0	0
- La personne clique sur "fin" par inadvertance	0	0	0	0	0	0
- La personne comprend que l'icône "Correction du porte-monnaie" correspond à la correction de la somme entrée en début de session						
- oui grâce au dessin du porte-monnaie	0	0	0	0	0	0
- oui grâce à la couleur jaune	0	0	0	0	0	0
- oui grâce au texte	0	0	0	0	0	0
- non	0	0	0	0	0	0
Entrée du poste budgétaire d'une dépense (10)						
- La personne met presque toutes les dépenses dans le poste "divers"	0	0	0	0	0	0
Entrée du poste budgétaire d'une recette (P.5) **ou d'une dépense** (P.10)						
- La personne se base sur les dessins pour se retrouver						
- uniquement	0	0	0	0	0	0
- en partie	0	0	0	0	0	0
- pas du tout	0	0	0	0	0	0
- La personne se base sur le texte pour se retrouver						
- uniquement	0	0	0	0	0	0
- en partie	0	0	0	0	0	0
- pas du tout	0	0	0	0	0	0
- La personne peut classer les recettes et les dépenses dans des postes comme "vêtements", "loisirs", "transports",...	0	0	0	0	0	0

Entrée du jour d'une recette ou d'une dépense (cycle d'un mois)							
- Pour le calendrier, la personne se base sur la couleur pour percevoir les jours passés et à venir	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- La personne se base sur le texte pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- Pour cette personne, la possibilité de gérer son budget au mois est :	- indispensable	0	0	0	0	0	0
	- très intéressante	0	0	0	0	0	0
	- intéressante	0	0	0	0	0	0
	- peu intéressante	0	0	0	0	0	0
	- sans intérêt	0	0	0	0	0	0
- La personne comprend qu'un calendrier est dessiné sur l'écran		0	0	0	0	0	0
- La personne est dérangée par le mois précédent		0	0	0	0	0	0
Entrée du jour d'une recette (p.6) **ou d'une dépense** (p.11) **(cycle d'une semaine)**							
- Pour la réglette du temps, la personne se base sur la couleur pour percevoir les jours passés et à venir	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- La personne se base sur le texte pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0

Entrée du moment d'une recette (p.7) **ou d'une dépense** (p.12) **(cycle d'une journée)**						
- La personne se base sur les dessins pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- La personne se base sur le texte pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
Etat du porte monnaie (représentation graphique) (p.15)						
- La personne comprend que la somme qu'elle voit est celle qu'elle a communiquée au programme au début de l'exercice		0	0	0	0	0	0
- La personne comprend qu'elle doit cliquer sur "SUITE" pour passer à l'écran suivant		0	0	0	0	0	0
Etat du porte monnaie (représentation par nombre) (P.16)						
- La personne comprend que la somme qu'elle voit est celle qu'elle a communiquée au programme au début de l'exercice		0	0	0	0	0	0

Récapitulatif des recettes (P.17) **et des dépenses** (p.18)						
- La personne comprend la signification des colonnes : - jour	0	0	0	0	0	0
- poste	0	0	0	0	0	0
- montant	0	0	0	0	0	0
- total	0	0	0	0	0	0
- La personne comprend la ligne "ce qu'il vous restait :"	0	0	0	0	0	0
- La personne comprend que les différentes lignes représentent les différentes recettes et dépenses qu'elle a enregistrées	0	0	0	0	0	0
- La personne comprend la signification des couleurs	0	0	0	0	0	0
Réglettes de couleur						
- La personne comprend que la réglette verte correspond aux recettes et la rouge aux dépenses	0	0	0	0	0	0
- La personne comprend que la réglette jaune correspond à l'état de son porte-monnaie	0	0	0	0	0	0
- La personne comprend que la grandeur des réglettes dépand des montants d'argent	0	0	0	0	0	0
- La personne comprend que la 3° réglette est égale à la différence entre la 1° réglette (recettes) et la 2° (dépenses)	0	0	0	0	0	0
- La personne comprend que pour avoir l'équilibre des comptes, il faut que la 3° réglette soit égale à la 4° réglette (Porte-monnaie)	0	0	0	0	0	0
- Les icônes "Recette","Dépense" et "Porte-monnaie" sont utiles à la personne	0	0	0	0	0	0
- Les montants d'argent totaux sont utils à la personne	0	0	0	0	0	0
- La personne comprend qu'elle doit cliquer sur "suite" pour passer à l'écran suivant	0	0	0	0	0	0

Généralités (pages 1 à 19)							
- Pour cette personne, les demandes de confirmation sont :	- indispensables	0	0	0	0	0	0
	- utiles	0	0	0	0	0	0
	- inutiles	0	0	0	0	0	0.
- Il arrive que la personne place mal le suiveur de la souris à cause de la forme de celui-ci		0	0	0	0	0	0
- La personne clique sur des objets non-interactifs :							
	- souvent	0	0	0	0	0	0
	- parfois	0	0	0	0	0	0
	- jamais	0	0	0	0	0	0
- En général, la personne se base sur les couleurs pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- En général, la personne se base sur le texte pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- En général, la personne se base sur les dessins pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- En général, la personne se base sur la forme des icônes pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- En général, la personne se base sur le son pour se retrouver	- uniquement	0	0	0	0	0	0
	- en partie	0	0	0	0	0	0
	- pas du tout	0	0	0	0	0	0
- L'accent anglais est très gênant pour la compréhension		0	0	0	0	0	0
- Une icône "oreille" serait utile dans tous les écrans		0	0	0	0	0	0
- La personne comprend que la couleur jaune correspond à la somme initiale		0	0	0	0	0	0
- La personne comprend que les couleurs verte et rouge correspondent aux recettes et dépenses		0	0	0	0	0	0
- La personne perçoit que le bord des écrans est jaune si l'écran est relatif au porte-monnaie, rouge pour les dépenses et vert pour les recettes		0	0	0	0	0	0
- La succession des écrans semble poser des problèmes à la personne		0	0	0	0	0	0

Annexe 3

Questionnaire sur le questionnaire

Questionnaire sur le questionnaire

- Combien de temps en moyenne, vous a-t-il fallu pour remplir ce questionnaire pour une séance : ____ minutes.

- Ce questionnaire est :
  - beaucoup trop fastidieux à remplir O
  - moyennement fastidieux O
  - pas fastidieux du tout O

- Selon vous, y a-t-il des questions inutiles
  - Oui, beaucoup O
  - Oui, un peu O
  - Non O
  - Si oui, lesquelles et pourquoi ?

- Selon vous, faudrait-il ajouter des questions ?
  - Oui O
  - Non O
  - Si oui, lesquelles et pourquoi ?

- Selon vous, faudrait-il formuler autrement certaines questions ?
  - Oui O
  - Non O
  - Si oui, lesquelles et pourquoi ?

- Selon vous, faudrait-il pouvoir répondre à certaines questions de manière plus nuancée ?
  - Oui O
  - Non O

Si oui, auxquelles et pourquoi ?

Annexe 4

Mode d'emploi "technique" du logiciel Comptes

LOGICIEL "COMPTES"

Mode d'emploi "technique"

Cette partie s'adresse exclusivement aux éducateurs. Elle a pour but de leur fournir la suite des opérations qu'il est nécessaire d'effectuer avant de pouvoir utiliser le logiciel "Comptes". Les manipulations décrites ci-dessous devraient les aider à préparer les différentes disquettes "Param" qui sont propres à chaque personne handicapée et à lancer correctement le programme "Comptes".

1. Aux utilisateurs de l'Amiga 500

Votre ordinateur devra impérativement être muni de 2 lecteurs de disquettes : un lecteur interne "nommé" DF0: et un lecteur "externe" nommé DF1: .

1.1. Préparation des disquettes "Param"

Pour chaque personne handicapée, il est nécessaire de préparer une disquette nommée "Param" avant que celle-ci n'utilise le logiciel pour la première fois.
Voici la liste des opérations à effectuer pour cette préparation:

- Allumer l'ordinateur
- A la demande de l'ordinateur, introduire une disquette "Workbench" en DF0: (lecteur "interne")
- Insérer une disquette vierge en DF1: (lecteur "externe")
- Formater cette disquette, c'est-à-dire :
    * Cliquer "gauche" une seule fois sur l'icône représentant la disquette vierge (DF1:BAD)
    * Cliquer "droit" pour obtenir la barre de menu
    * Choisir dans le menu "DISK" et puis l'option "Initialize"

- Une fois la disquette formatée, il faut la renommer c'est-à-dire :
  * Cliquer "gauche" sur l'icône représentant la disquette vierge et formatée (Empty)
  * Cliquer "droit" pour obtenir la barre de menu
  * Choisir dans le menu "Workbench" et puis l'option "Rename"
  * Remplacer le mot "Empty" apparaissant par "Param". Attention, il est important de taper un "P" majuscule et de taper "aram" en minuscules !
- Retirer la disquette qui s'appelle à présent "Param" du lecteur DF1: . Pour éviter tout risque de confusion, nous vous conseillons d'écrire sur l'étiquette autocollante de cette disquette : "Param" suivi du nom de la personne handicapée concernée.
- Vous pouvez à présent passer au paragraphe 1.2.

1.2. Définition des paramètres

Cette opération doit se faire impérativement avant que la personne handicapée n'utilise le logiciel pour la première fois. Mais elle peut également être entreprise entre 2 séances d'utilisation du programme si l'éducateur désire modifier l'une ou l'autre des valeurs affectées aux paramètres précédemment.

Dans les 2 cas, suivez la procédure suivante :

- Introduire la disquette "Workbench" en DF0: et la disquette "Comptes" en DF1:
- Double cliquer "gauche" sur la disquette "Comptes"
- Double cliquer "gauche" sur le fichier "Paramètres"
- Fixer la valeur des paramètres de travail en fonction de la personne handicapée pour laquelle ils sont destinés
- A la demande de l'ordinateur, retirer la disquette "Comptes" et introduire la disquette "Param" en DF1:
- Une fois cette opération terminée, retirer les 2 disquettes de leurs unités et éteindre l'ordinateur

Vous pouvez maintenant commencer à utiliser le programme "Comptes".(point 1.3.)

1.3. Exécution du programme "Comptes"

- Insérer la disquette "Comptes" en DF0: et la disquette "Param" en DF1:
- Allumer l'ordinateur. Après quelques instants, la phrase "Quel est votre nom ?" s'affiche à l'écran. Il faut alors laisser la place à la personne handicapée car le programme est déjà en route
- Attention, il faut laisser la disquette "Param" en DF1: jusqu'à la complète terminaison du programme.

2. Aux utilisateurs de l'Amiga 2000 (version auto-boot)

Votre ordinateur devra impérativement être muni d'un disque dur et d'un lecteur de disquette interne "nommé" DF0: .

2.1. Préparation du disque dur

Cette opération doit être réalisée une fois pour toutes. Elle permet l'installation du programme "Comptes" sur votre disque dur.

- Allumer l'ordinateur et charger le système
- Insérer la disquette "Comptes" en DF0:
- Double-cliquer "gauche" sur la disquette "Comptes"
- Tirer le fichier "Comptes" sur la disquette "Workbench"
- Tirer le fichier "Paramètres" sur la disquette "Workbench"
- Fermer la fenêtre "Comptes".

2.2. Préparation des disquettes "Param"

Pour chaque personne handicapée, il est nécessaire de préparer une disquette nommée "Param" avant que celle-ci n'utilise le logiciel pour la première fois.
Voici la liste des opérations à effectuer pour cette préparation:

- Allumer l'ordinateur
- Insérer une disquette vierge en DF0:
- Formater cette disquette, c'est-à-dire :
    * Cliquer "gauche" une seule fois sur l'icône représentant la disquette vierge (DF0:BAD)
    * Cliquer "droit" pour obtenir la barre de menu
    * Choisir dans le menu "DISK" et puis l'option "Initialize"

- Une fois la disquette formatée, il faut la renommer c'est-à-dire :
    * Cliquer "gauche" sur l'icône représentant la disquette vierge et formatée (Empty)
    * Cliquer "droit" pour obtenir la barre de menu
    * Choisir dans le menu "Workbench" et puis l'option "Rename"
    * Remplacer le mot "Empty" apparaissant par "Param". Attention, il est important de taper un "P" majuscule et de taper "aram" en minuscules !
- Retirer la disquette qui s'appelle à présent "Param" du lecteur DF0: . Pour éviter tout risque de confusion, nous vous conseillons d'écrire sur l'étiquette autocollante de cette disquette : "Param" suivi du nom de la personne handicapée concernée.
- Vous pouvez à présent passer au paragraphe 2.3.

2.3. Définition des paramètres

Cette opération doit se faire impérativement avant que la personne handicapée n'utilise le logiciel pour la première fois. Mais elle peut également être entreprise entre 2 séances d'utilisation du programme si l'éducateur désire modifier l'une ou l'autre des valeurs affectées aux paramètres précédemment.

Dans les 2 cas, suivez la procédure suivante :

- Allumer l'ordinateur
- Introduire la disquette "Param" en DF0:
- Double cliquer "gauche" sur la disquette "Workbench"
- Double cliquer "gauche" sur le fichier "Paramètres"
- Fixer la valeur des paramètres de travail en fonction de la personne handicapée pour laquelle ils sont destinés
- Une fois cette opération complètement terminée, retirer la disquette de son unité et éteindre l'ordinateur

Vous pouvez maintenant commencer à utiliser le programme "Comptes". (point 2.4.)

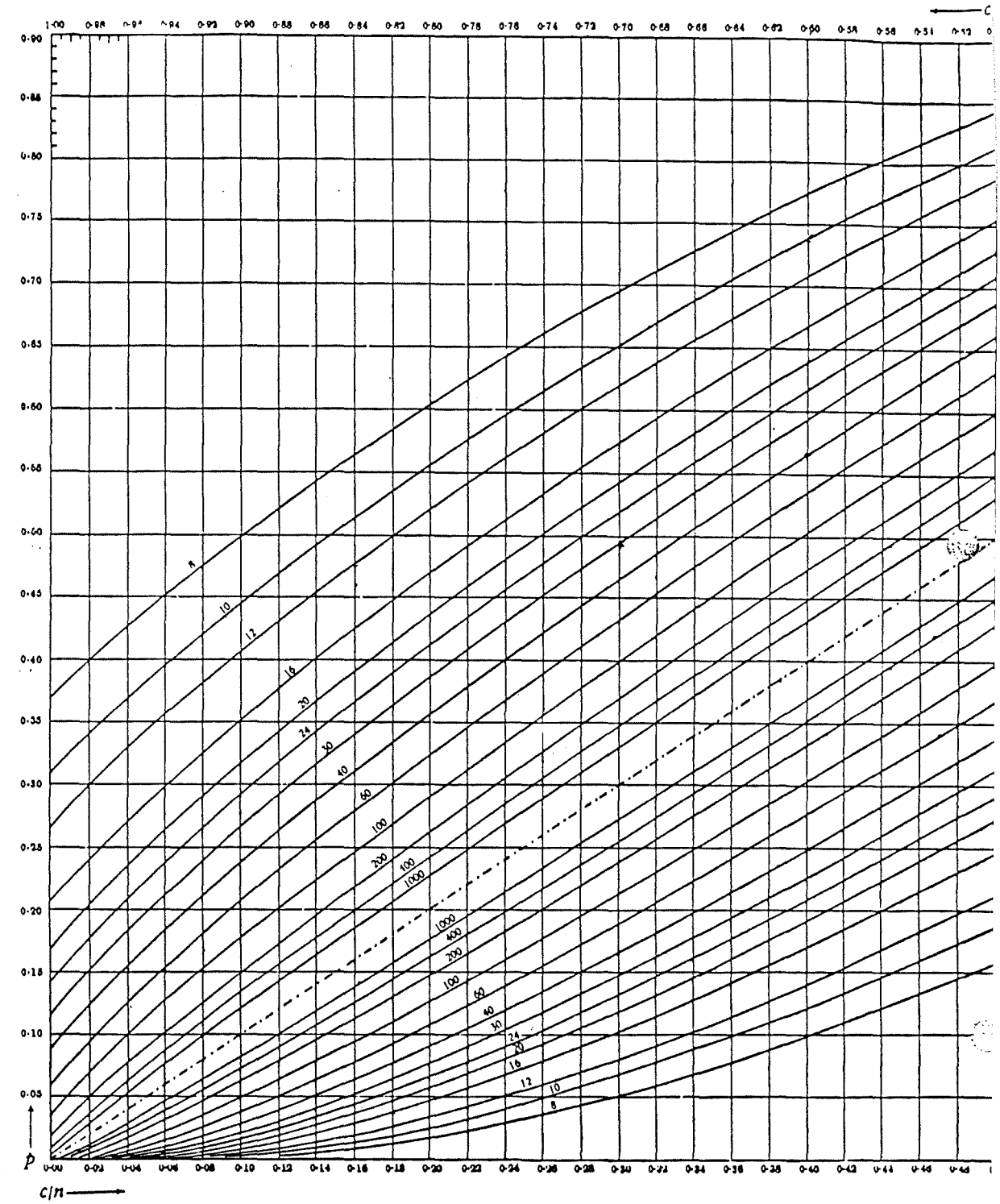
2.4. Exécution du programme "Comptes"

- Insérer la disquette "Param" de l'utilisateur concerné en DF0:
- Allumer l'ordinateur.
- Double-cliquer "gauche" sur "Workbench"
- Double-cliquer "gauche" sur le fichier "Comptes"
- Après quelques instants, la phrase "Quel est votre nom ?" s'affiche à l'écran. Il faut alors laisser la place à la personne handicapée car le programme est déjà en route.
- Attention, il faut laisser la disquette "Param" en DF0: jusqu'à la complète terminaison du programme.

Annexe 5

Table de la distribution binomiale

Table 41. *Chart providing confidence limits for p in binomial sampling, given a sample fraction c/*
Confidence coefficient, $1 - 2\alpha = 0{\cdot}95$.



The numbers printed along the curves indicate the sample size n. If for a given value of the abscissa c/n, p_A a
are the ordinates read from (or interpolated between) the appropriate lower and upper curves, then

$$\Pr\{p_A \leqslant p \leqslant p_B\} \geqslant 1 - 2\alpha.$$

Annexe 6

Présentation des fréquences et intervalles de confiance calculés à partir des réponses au questionnaire

Nous présentons les fréquences et intervalles de confiance relatifs à certaines questions que nous avons utilisées pour répondre aux objectifs de l'évaluation.

Ces informations se présentent sous la forme d'un tableau :

- la première colonne indique le numéro de la question et il est possible d'accéder à son énoncé en annexe 1;

- la deuxième colonne indique le nombre de réponses sur lesquelles nous avons pu nous baser, c'est-à-dire l'effectif de l'échantillon;

- la troisième colonne indique le nombre de succès ou de progrès rencontrés dans l'échantillon;

- la quatrième colonne indique la fréquence de succès ou de progrès;

- la cinquième colonne fournit la limite inférieure de l'intervalle de confiance obtenu pour cette fréquence;

- enfin, la dernière colonne fournit la limite supérieure de cet intervalle.

5) Le logiciel permet-il d'apprendre de nouveaux termes?

Question	Effectif	Nombre de progrès	Fréquence de progrès	limite inférieure	limite supérieure
E18	10	6	60%	0,27	0,88
E23	19	0	0%	0	0,17
E26	10	0	0%	0	0,31
P32	13	4	31%	0,09	0,62
P33	6	3	50%	0,12	0,87

8) Le logiciel peut-il entraîner des changements dans les comportements des individus vis-à-vis de l'argent?

Qest°	effectif	nbre de progrès	fréq. de progrès	limite inf.	limite sup.
P40	9	1	11%	0	0,47
P41	9	0	0%	0	0,35
P42	13	3	23%	0,05	0,53
P43	3	2	66%	-	-
P44	7	1	14%	0	0,67
P45	15	6	40%	0,16	0,67
P46	11	0	0%	-	-
P47	8	4	50%	0,16	0,84
P27	5	0	0%	-	-
P28	11	0	0%	-	-
P29	6	1	16%	0	0,63
P30	10	0	0%	-	-
P31	10	0	0%	-	-

9) Le logiciel permet-il aux personnes handicapées mentales d'acquérir une meilleure notion des valeurs?

Qest°	effectif	nbre de progrès	fréq. de progrès	limite inf.	limite sup.
P25	8	0	0%	-	-
P36	2	0	0%	-	-
P37	12	2	16%	0,02	0,47
P38	12	0	0%	-	-
P39	17	0	0%	-	-
P34	8	2	25%	0,03	0,65
P35	16	5	31%	0,11	0,58

11) L'interface réalisée dans le logiciel convient-elle bien aux personnes handicapées mentales ou devrait-elle être améliorée?

Quest°	effectif	nbre de succès	freq. de succès	limite inf.	limite sup.
souris					
E10	20	17	85%	0,63	0,97
E20	20	20	100%	0,83	1
E45	20	14	70%	0,46	0,88
clavier					
E1	20	17	85%	0,63	0,97
E15	16	13	81%	0,54	0,96
E17	18	7	39%	0,17	0,64
termes					
E2	20	11	55%	0,32	0,77
E3	20	13	65%	0,42	0,85
E13	20	14	70%	0,46	0,88
E18	20	16	80%	0,57	0,97
E31	20	18	90%	0,68	0,99
E33	20	13	65%	0,42	0,85
E34	20	13	65%	0,42	0,85
E35	20	15	75%	0,52	0,93
E36	20	13	65%	0,42	0,85
E37	20	12	60%	0,37	0,8
E42	20	10	50%	0,27	0,73
dessins					
E5	20	20	100%	0,83	1
E6	20	20	100%	0,83	1
E11	20	15	75%	0,52	0,93
E19	20	20	100%	0,83	1
son					
E51	20	11	55%	0,32	0,77
Utilité					
E47	20	15	75%	0,52	0,93
E48	20	16	80%	0,57	0,97
E49	20	20	100%	0,83	1
E50	20	12	60%	0,37	0,8
thermom.					
E30	20	1	5%	0	0,25
E39	20	8	40%	0,19	0,64
E40	20	3	15%	0,03	0,37
E41	20	2	10%	0,01	0,32
E53	20	1	5%	0	0,25
E54	20	4	20%	0,06	0,43
E55	20	5	25%	0,08	0,49
E56	20	8	40%	0,19	0,64
E58	20	1	5%	0	0,25

(suite du tableau précédent)

Qest°	effectif	nbre de succès	freq. de succès	limite inf.	limite sup.
balance					
E42	20	10	50%	0,27	0.73
E43	20	11	55%	0,32	0,77

Annexe 7

Table de la distribution t de Student

TABLE DE LA v. a. DE STUDENT

k \ a	0,90	0,95	0,975	0,99	0,995
1	3,08	6,31	12,71	31,82	63,66
2	1,89	2,92	4,30	6,96	9,93
3	1,64	2,35	3,18	4,54	5,84
4	1,53	2,13	2,78	3,75	4,60
5	1,48	2,01	2,57	3,36	4,03
6	1,44	1,94	2,45	3,14	3,71
7	1,42	1,90	2,36	3,00	3,50
8	1,40	1,86	2,31	2,90	3,36
9	1,38	1,83	2,26	2,82	3,25
10	1,37	1,81	2,23	2,76	3,17
11	1,36	1,80	2,20	2,72	3,11
12	1,36	1,78	2,18	2,68	3,06
13	1,35	1,77	2,16	2,65	3,01
14	1,34	1,76	2,14	2,62	2,98
15	1,34	1,75	2,13	2,60	2,95
16	1,34	1,75	2,12	2,58	2,92
17	1,33	1,74	2,11	2,57	2,90
18	1,33	1,73	2,10	2,55	2,88
19	1,33	1,73	2,09	2,54	2,86
20	1,32	1,72	2,09	2,53	2,85
21	1,32	1,72	2,08	2,52	2,83
22	1,32	1,72	2,07	2,51	2,82
23	1,32	1,71	2,07	2,50	2,81
24	1,32	1,71	2,06	2,49	2,80
25	1,32	1,71	2,06	2,48	2,79
26	1,32	1,71	2,06	2,48	2,78
27	1,31	1,70	2,05	2,47	2,77
28	1,31	1,70	2,05	2,47	2,76
29	1,31	1,70	2,04	2,46	2,76
30	1,31	1,70	2,04	2,46	2,75
50	1,30	1,68	2,01	2,40	2,68
100	1,29	1,66	1,98	2,37	2,63
$+\infty$	1,28	1,65	1,96	2,33	2,58

Fournit la valeur $t_{k,a}$ telle que $P\left[t_k \leq t_{k,a}\right] = a$.

Annexe 8

Présentation des résultats obtenus grâce à l'analyse en composantes principales (Listing fourni par le logiciel SAS)

Principal Component Analysis

Première analyse en composantes principales

20 Observations
11 Variables

Simple Statistics

	Q4	Q9	Q18	Q25	Q26	Q28
Mean	0.5000000000	0.5000000000	0.2500000000	0.6250000000	0.5750000000	0.4500000000
StD	0.5129891760	0.5129891760	0.4442616583	0.4832728335	0.4666510022	0.5104177855

	Q35	Q42	E10	E20	E46
Mean	0.5000000000	0.3500000000	0.7500000000	0.9000000000	0.6750000000
StD	0.3244428423	0.4893604849	0.4442616583	0.3077935056	0.3354101966

Correlation Matrix (Matrice des corrélations)

	Q4	Q9	Q18	Q25	Q26	Q28	Q35	Q42	E10	E20	E46
Q4	1.0000	0.8000	0.5774	0.3715	0.2748	0.1005	0.6325	0.1048	0.1155	0.3333	0.2294
Q9	0.8000	1.0000	0.5774	0.1592	0.2748	0.1005	0.6325	0.1048	0.1155	0.3333	0.2294
Q18	0.5774	0.5774	1.0000	0.4596	0.5395	0.4062	0.5477	0.5447	0.0667	0.1925	0.2208
Q25	0.3715	0.1592	0.4596	1.0000	0.7147	0.5067	0.3357	0.5842	-.2145	-.2654	0.1826
Q26	0.2748	0.2748	0.5395	0.7147	1.0000	0.5137	0.2607	0.6857	-.1587	-.3115	0.3321
Q28	0.1005	0.1005	0.4062	0.5067	0.5137	1.0000	0.1589	0.8112	-.4062	-.3685	-.0231
Q35	0.6325	0.6325	0.5477	0.3357	0.2607	0.1589	1.0000	0.1657	0.3651	0.5270	0.3627
Q42	0.1048	0.1048	0.5447	0.5842	0.6857	0.8112	0.1657	1.0000	-.3026	-.4543	-.0721
E10	0.1155	0.1155	0.0667	-.2145	-.1587	-.4062	0.3651	-.3026	1.0000	0.5774	0.4857
E20	0.3333	0.3333	0.1925	-.2654	-.3115	-.3685	0.5270	-.4543	0.5774	1.0000	0.1784
E46	0.2294	0.2294	0.2208	0.1826	0.3321	-.0231	0.3627	-.0721	0.4857	0.1784	1.0000

Eigenvalues of the Correlation Matrix (Valeurs propres)

	Eigenvalue	Difference	Proportion	Cumulative
PRIN1	4.10787	1.01282	0.373443	0.37344
PRIN2	3.09505	1.90356	0.281368	0.65481
PRIN3	1.19149	0.45909	0.108317	0.76313
PRIN4	0.73240	0.21906	0.066582	0.82971
PRIN5	0.51334	0.08537	0.046667	0.87638
PRIN6	0.42797	0.11711	0.038907	0.91528
PRIN7	0.31086	0.04933	0.028260	0.94354
PRIN8	0.26153	0.07378	0.023775	0.96732
PRIN9	0.18774	0.07558	0.017068	0.98439
PRIN10	0.11217	0.05258	0.010197	0.99458
PRIN11	0.05958	.	0.005417	1.00000

Principal Component Analysis

Eigenvectors (Vecteurs propres)

	PRIN1	PRIN2	PRIN3	PRIN4	PRIN5	PRIN6	PRIN7	PRIN8	PRIN9	PRIN10	PRIN11
Q4	0.333039	0.265982	-.314347	-.362356	0.067300	0.027325	0.198228	0.458792	0.239587	-.523242	-.044492
Q9	0.313213	0.278936	-.349514	-.276330	-.410146	0.041920	0.198547	-.257418	0.073283	0.560044	0.175763
Q18	0.408966	0.083952	-.102962	0.241565	-.195485	-.506417	-.414447	0.283098	-.386798	0.079853	-.232747
Q25	0.364941	-.176645	0.164242	-.215735	0.704688	-.015673	-.000167	0.207563	-.009601	0.448185	0.145681
Q26	0.383876	-.180519	0.293304	-.197665	-.017749	-.309118	-.088229	-.549598	0.401019	-.212334	-.288687
Q28	0.304900	-.317730	-.040966	0.377914	-.191212	0.560419	-.066199	0.226256	0.337714	0.157108	-.341535
Q35	0.325530	0.321303	-.040134	0.219083	0.281551	0.380216	0.168290	-.411409	-.492893	-.208112	-.183079
Q42	0.342082	-.330961	0.027339	0.374981	-.135531	-.129080	0.289367	-.027670	-.031051	-.237023	0.674843
E10	-.021442	0.418272	0.437341	0.334565	-.014813	-.251540	0.557575	0.192984	0.197333	0.177188	-.198565
E20	-.000532	0.487255	-.119447	0.362441	0.241882	0.014759	-.482763	-.121304	0.459393	-.005715	0.315241
E46	0.163626	0.233491	0.667721	-.271384	-.314974	0.326271	-.285214	0.157446	-.136987	-.068079	0.253002

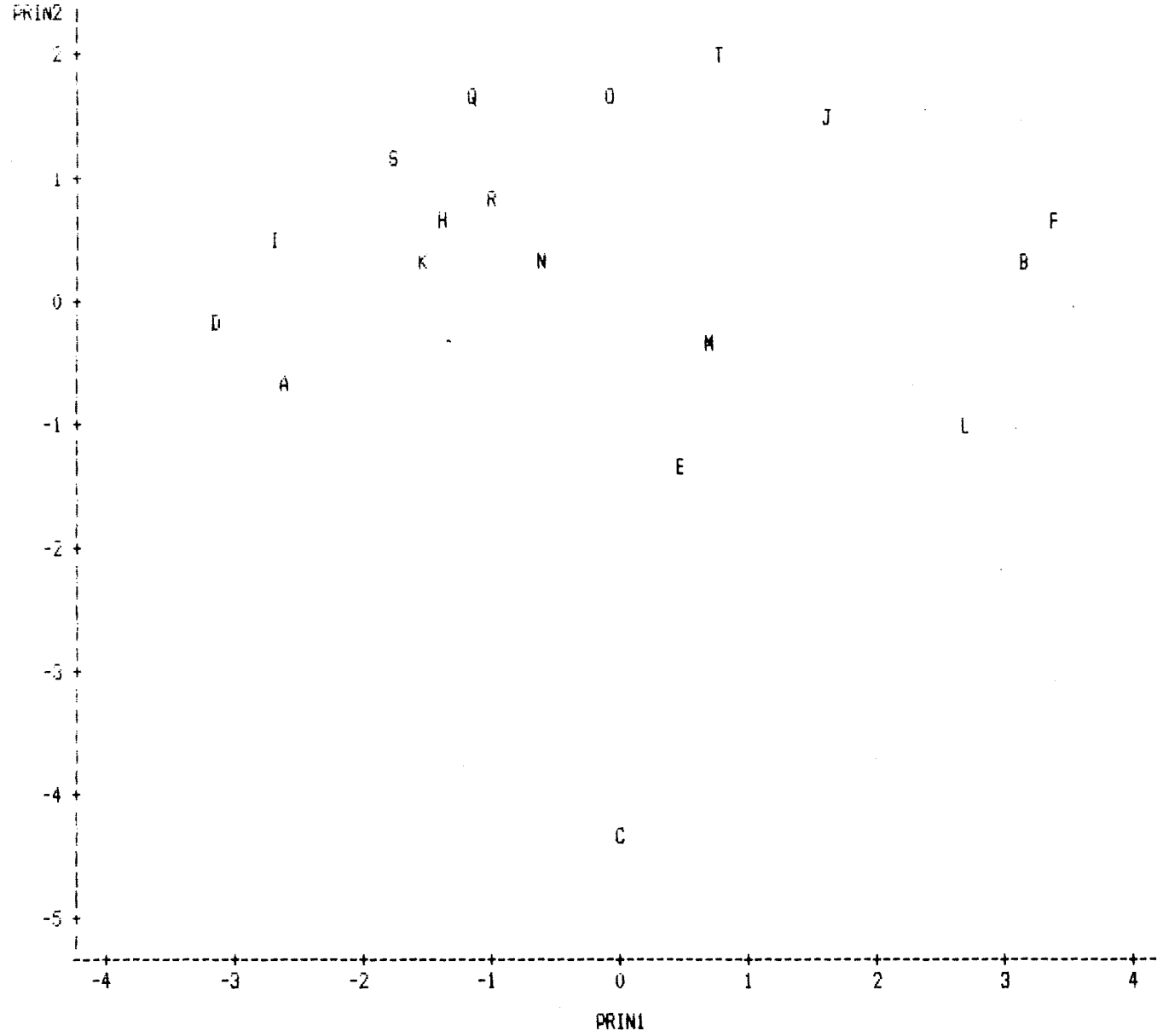
Projection des individus sur les nouveaux axes

PERSONNE	PRIN1	PRIN2	PRIN3	PRIN4	PRIN5
D	-3.16167	-0.18202	-0.92687	0.92231	0.09043
I	-2.67384	0.51411	1.06389	0.11320	-0.84864
A	-2.61173	-0.62836	-1.97314	0.50686	0.55767
S	-1.80552	1.20495	-0.67467	0.31672	-0.74473
K	-1.57481	0.38678	0.92178	1.19123	-0.78936
H	-1.38328	0.63309	1.48623	0.01584	0.29532
Q	-1.15630	1.72345	-1.28744	-0.38964	-0.61354
R	-1.01172	0.81418	-0.26626	-0.29738	1.64414
N	-0.59440	0.25692	1.97042	-0.41916	1.00538
O	-0.08977	1.68468	0.33646	-1.21778	-1.12111
C	0.00639	-4.40967	0.41615	-0.77619	-0.36307
G	0.00639	-4.40967	0.41615	-0.77619	-0.36307
E	0.45807	-1.38996	0.95065	1.49207	0.82334
M	0.65577	-0.39948	-1.69801	-1.06052	0.48433
T	0.75574	2.00774	0.30020	-1.11476	0.78996
J	1.58593	1.50813	0.44456	-1.12044	-0.10297
L	2.68667	-1.08024	-1.55964	0.03770	-0.25167
B	3.14008	0.35642	-0.63707	1.12841	0.14889
F	3.38400	0.70448	0.35831	0.72386	-0.32065
P	3.38400	0.70448	0.35831	0.72386	-0.32065

PLOT OF THE 1 ET 2 PRINCIPAL COMPONENTS

Représentation graphique des individus sur les axes

Plot of PRIN2*PRIN1. Symbol is value of PERSONNE.



NOTE: 2 obs hidden.

Principal Component Analysis

20 Observations
10 Variables

Deuxième analyse en composantes principales

Simple Statistics

	Q4	Q9	Q18	Q25	Q26
Mean	0.5000000000	0.5000000000	0.2500000000	0.6250000000	0.5750000000
StD	0.5129891760	0.5129891760	0.4442616583	0.4832728335	0.4666510022

	Q28	Q35	Q42	E47	E48
Mean	0.4500000000	0.5000000000	0.3500000000	0.4000000000	0.4500000000
StD	0.5104177655	0.3244428423	0.4893604849	0.2615741819	0.2762531257

Correlation Matrix

	Q4	Q9	Q18	Q25	Q26	Q28	Q35	Q42	E47	E48
Q4	1.0000	0.8000	0.5774	0.3715	0.2748	0.1005	0.6325	0.1048	-.3922	0.5571
Q9	0.8000	1.0000	0.5774	0.1592	0.2748	0.1005	0.6325	0.1048	-.3922	0.5571
Q18	0.5774	0.5774	1.0000	0.4596	0.5395	0.4062	0.5477	0.5447	-.2265	0.5361
Q25	0.3715	0.1592	0.4596	1.0000	0.7147	0.5067	0.3357	0.5842	-.1041	0.6406
Q26	0.2748	0.2748	0.5395	0.7147	1.0000	0.5137	0.2607	0.6857	0.0647	0.6430
Q28	0.1005	0.1005	0.4062	0.5067	0.5137	1.0000	0.1589	0.8112	-.0394	0.3546
Q35	0.6325	0.6325	0.5477	0.3357	0.2607	0.1589	1.0000	0.1657	-.4651	0.5872
Q42	0.1048	0.1048	0.5447	0.5842	0.6857	0.8112	0.1657	1.0000	-.1234	0.5256
E47	-.3922	-.3922	-.2265	-.1041	0.0647	-.0394	-.4651	-.1234	1.0000	-.6191
E48	0.5571	0.5571	0.5361	0.6406	0.6430	0.3546	0.5872	0.5256	-.6191	1.0000

Eigenvalues of the Correlation Matrix

	Eigenvalue	Difference	Proportion	Cumulative
PRIN1	4.84005	2.67577	0.484005	0.48401
PRIN2	2.16428	1.25252	0.216428	0.70043
PRIN3	0.91176	0.22806	0.091176	0.79161
PRIN4	0.68370	0.26257	0.068370	0.85998
PRIN5	0.42114	0.04255	0.042114	0.90209
PRIN6	0.37858	0.05579	0.037858	0.93995
PRIN7	0.32279	0.19843	0.032279	0.97223
PRIN8	0.12436	0.02924	0.012436	0.98467
PRIN9	0.09512	0.03691	0.009512	0.99418
PRIN10	0.05821	.	0.005821	1.00000

Principal Component Analysis

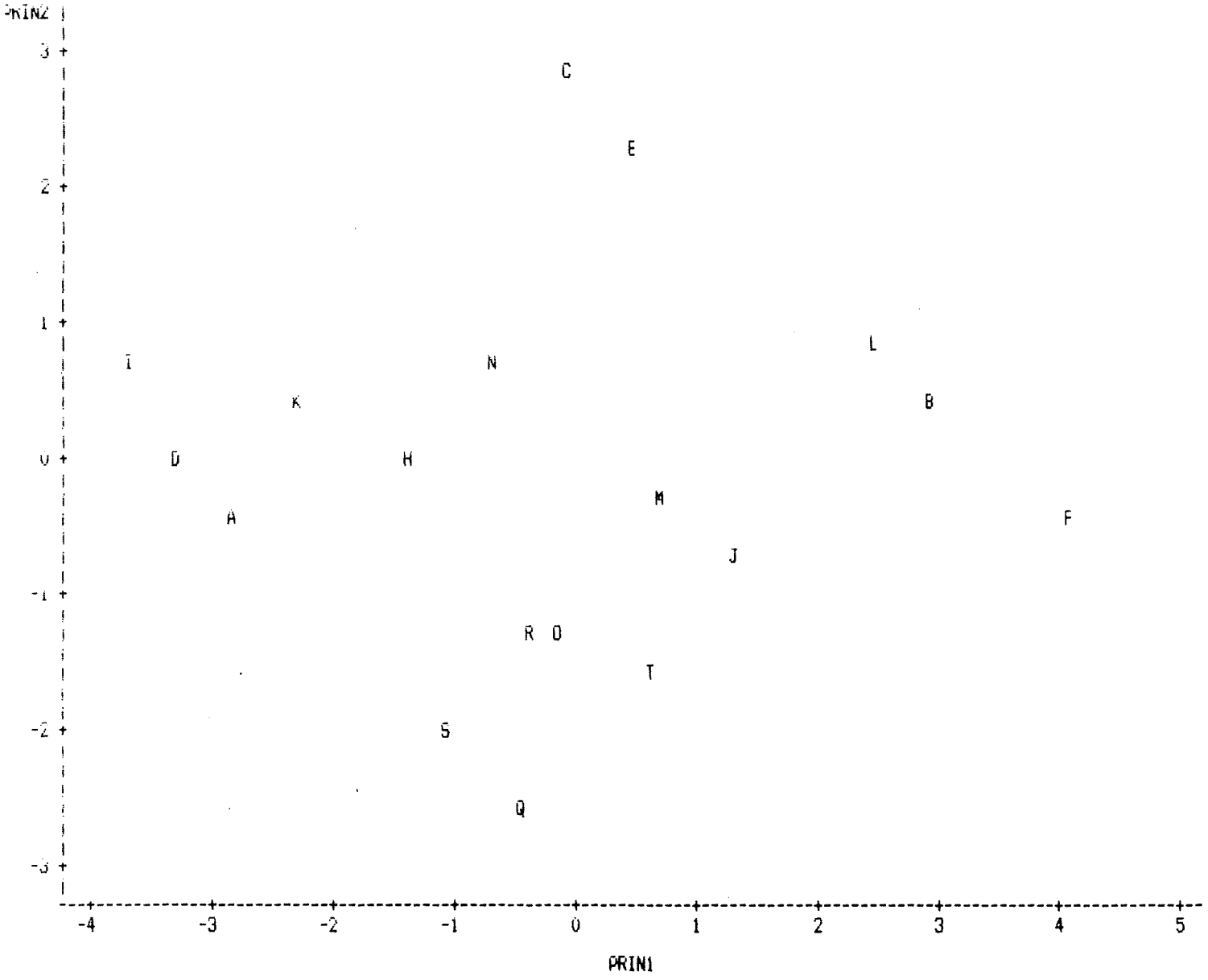
Eigenvectors

	PRIN1	PRIN2	PRIN3	PRIN4	PRIN5	PRIN6	PRIN7	PRIN8	PRIN9	PRIN10
Q4	0.318330	-.351634	0.283907	-.030255	-.025397	0.509295	-.287041	0.561910	-.175173	0.067183
Q9	0.303313	-.379916	0.279972	0.202130	-.355062	0.202615	0.331789	-.428696	0.376597	-.203993
Q18	0.362240	-.015153	0.290385	0.336198	-.045192	-.516119	-.551369	-.229219	-.178629	0.119674
Q25	0.328212	0.259828	0.013193	-.529468	0.343390	0.257976	-.357935	-.352933	0.272588	-.181119
Q26	0.330244	0.321345	0.224055	-.342950	-.300827	-.226113	0.331231	0.162524	-.415872	-.411289
Q28	0.258593	0.406218	-.138622	0.502611	0.210920	0.450585	0.193109	-.227356	-.389860	0.088298
Q35	0.314429	-.311392	0.022645	0.036902	0.724578	-.310526	0.391011	0.152713	0.051166	-.045006
Q42	0.306361	0.420589	-.184117	0.293815	-.134575	-.099390	-.025141	0.464451	0.600726	-.061755
E47	-.202844	0.343588	0.747169	-.051027	0.128843	0.033717	0.200420	0.031587	0.179908	0.434357
E48	0.397401	-.062836	-.307286	-.323013	-.242111	-.070986	0.186288	-.072356	-.021901	0.731633

PERSONNE	PRIN1	PRIN2	PRIN3	PRIN4	PRIN5
I	-3.68542	0.70119	1.42726	0.44960	-0.19359
D	-3.29769	0.04442	-0.00096	0.54714	-0.43987
A	-2.81312	-0.43547	0.03394	0.60401	0.67678
K	-2.30649	0.36038	-0.23764	1.58872	1.09001
H	-1.40043	0.06393	-0.26851	-0.89588	0.27152
S	-1.11485	-1.94656	-1.40468	0.51094	-0.69986
N	-0.70702	0.67706	-0.01479	-1.81113	0.30447
Q	-0.49431	-2.63241	-0.85124	0.45196	-0.74936
R	-0.40643	-1.35418	-1.36971	-1.03765	0.65333
O	-0.17436	-1.28702	1.05711	-0.38049	-1.14773
C	-0.05891	2.81227	-0.69751	-0.28289	-0.67395
G	-0.05891	2.81227	-0.69751	-0.28289	-0.67395
E	0.42566	2.33238	-0.66262	-0.22602	0.44270
T	0.63551	-1.57358	0.87924	-1.05175	1.00179
M	0.65757	-0.29784	0.57276	-0.12392	0.29837
J	1.32016	-0.78349	1.73805	-0.71933	-0.53891
L	2.45283	0.87183	1.09022	0.86578	-0.40068
B	2.93740	0.39194	1.12512	0.92265	0.71597
F	4.04441	-0.37856	-0.85927	0.43556	0.03148
P	4.04441	-0.37856	-0.85927	0.43556	0.03148

PLOT OF THE 1 ET 2 PRINCIPAL COMPONENTS

Plot of PRIN2*PRIN1. Symbol is value of PERSONNE.



NOTE: 2 obs hidden.

Annexe 9

Listing de la seconde version du logiciel Comptes

- Le programme Comptes
- Le programme Paramètres
- Le programme d'aide à l'évaluation du logiciel Comptes

Le programme Comptes (version belge) : les "implémentations modules"

```
MODULE Coordinateur;
  FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
  FROM MathLib0 IMPORT entier,real;
  FROM SYSTEM IMPORT ADR,SHORT;
  FROM Conversions IMPORT ConvNumberToString;
  FROM Strings IMPORT Relation, CompareString;
  FROM Interface IMPORT SSaisieNom;
  FROM ModReglette IMPORT Reglette;
  FROM ModEquilibre IMPORT Equilibre;
  FROM ModPresentation IMPORT Presentation;
  FROM Interface IMPORT Menu, CorrectionPM;
  IMPORT EtatPorteMonnaie;
  FROM EnregistrerSomme IMPORT EnregistrerDepense, EnregistrerRecette,
                               EnregistrerSommeInitiale;
  FROM VariablesGlobales IMPORT Str20,Str40,Sinit, Sactuel, TableauParametres,
                                Nom,TRecettes, TDepenses, TableauTraces, cmap,
                                TabNombre,Max ;
  FROM VariablesGlobales IMPORT TempsSommeInit,TempsSIBillets,TempsCorrection,
       TempsSommeRec,TempsMomentRec,TempsPosteRec,TempsSommeDep,TempsMomentDep,
       TempsPosteDep,TempsTotal,TableauChaine,TableauTemps;
  FROM Strings IMPORT ConcatString, CopyString,OverwriteWithSubString,
                      StringLength;
  FROM InOut IMPORT OpenInputFile, CloseInput, OpenOutputFile, CloseOutput,
                    Done, ReadInt, WriteString, ReadString,WriteInt,WriteLn;
  FROM RealInOut IMPORT ReadReal, WriteReal;
  FROM RealConversions IMPORT ConvRealToString,RealToStringFormat;
  FROM Intuition IMPORT CloseScreen,ScreenPtr,Window,WindowPtr,WindowFlags,
       WindowFlagsSet,IDCMPFlags,IDCMPFlagsSet,CloseWindow,ScreenToFront;
  FROM SimpleWindows IMPORT CreateWindow;
  FROM SimpleScreens IMPORT CreateScreen;
  FROM Views IMPORT LoadRGB4;

  CONST
      WIDCMP = IDCMPFlagsSet {};
      WFlags = WindowFlagsSet {Borderless};

  VAR EtatSortie, Fonction,Reel,St1 : Str20;
      Ph : Str40;
      Scr : ScreenPtr;
      Win : WindowPtr;
      In : ARRAY [0..2] OF CHAR;
      DSR : DateStampRecord;
      DernVers,s : REAL;
      FileName,FileName2,FileName3 : ARRAY [0..40] OF CHAR;
      Montant : ARRAY [0..15] OF CHAR;
      Ligne : ARRAY [0..79] OF CHAR;
      Q,FinProg : BOOLEAN;
      LI1,LI2,Indice1,Indice2 : LONGINT;
      i : INTEGER;
      ent : INTEGER;
      ree : REAL;
      MinEND, TicksEND,MinTotBEG,TicksTotBEG : INTEGER;

  PROCEDURE Impression;
     VAR imp : INTEGER;
     BEGIN
        DateStamp (DSR);
        LI1 := 1D;
        LI2 := 7D;
        Indice1 := ((DSR.dsDays - LI1) MOD LI2);
        Indice2 := ((DSR.dsDays - LI1) MOD LI2);
        i := SHORT (Indice1);
        CASE i OF
          0 : Ligne := "      Lundi"|
          1 : Ligne := "      Mardi"|
          2 : Ligne := "      Mercredi"|
```

```
  3 : Ligne := "      Jeudi";
  4 : Ligne := "      Vendredi";
  5 : Ligne := "      Samedi";
  6 : Ligne := "      Dimanche"
END;
OpenOutputFile ("prt:");
WriteLn;
WriteString ("     ");
WriteString (Nom);
WriteString ("     ");
WriteString (Ligne);
WriteLn;
WriteLn;
WriteString ("     SOLDES DES COMPTES EN DEBUT DE SESSION : ");
ConvRealToString (TRecettes.Tab[0].Valeur,Montant,2,Decimal);
WriteString (Montant);
WriteLn;
WriteLn;
WriteString ("     RECETTES ENREGISTREES");
WriteLn;
WriteLn;
WriteString ("     MOMENT              POSTE                    MONTANT ")
WriteLn;
WriteLn;
CloseOutput();
imp := 1;
WHILE imp <= TRecettes.Indice-1 DO
  OpenOutputFile ("prt:");
  Ligne := "                                                              ";
  OverwriteWithSubString (Ligne,TRecettes.Tab[imp].Jour,5);
  OverwriteWithSubString (Ligne,TRecettes.Tab[imp].Poste,24);
  ConvRealToString (TRecettes.Tab[imp].Valeur,Montant,2,Decimal);
  OverwriteWithSubString (Ligne,Montant,53-StringLength(Montant));
  WriteString (Ligne);
  WriteLn;
  imp := imp + 1;
  CloseOutput();
END;
OpenOutputFile ("prt:");
WriteLn;
WriteString ("     DEPENSES ENREGISTREES");
WriteLn;
WriteLn;
WriteString ("     MOMENT              POSTE                    MONTANT "
WriteLn;
WriteLn;
CloseOutput();
imp := 0;
WHILE imp <= TDepenses.Indice-1 DO
  OpenOutputFile ("prt:");
  Ligne := "                                                              ";
  OverwriteWithSubString (Ligne,TDepenses.Tab[imp].Jour,5);
  OverwriteWithSubString (Ligne,TDepenses.Tab[imp].Poste,24);
  ConvRealToString (TDepenses.Tab[imp].Valeur,Montant,2,Decimal);
  OverwriteWithSubString (Ligne,Montant,53-StringLength(Montant));
  WriteString (Ligne);
  WriteLn;
  imp := imp + 1;
  CloseOutput();
END;
OpenOutputFile ("prt:");
WriteLn;
WriteString ("     SOLDES DES COMPTES EN FIN DE SESSION : ");
ConvRealToString (Sactuel,Montant,2,Decimal);
WriteString (Montant);
WriteLn;
```

```
      WriteLn;
      WriteString ("     ETAT DU PORTE-MONNAIE : ");
      ConvRealToString (Sinit,Montant,2,Decimal);
      WriteString (Montant);
      WriteLn;
      CloseOutput();
END Impression;

 PROCEDURE ImpApres;
   VAR imp : INTEGER;
   VAR FiNa : ARRAY [0..79] OF CHAR;
   BEGIN
      DateStamp (DSR);
      LI1 := 1D;
      LI2 := 7D;
      Indice1 := ((DSR.dsDays - LI1) MOD LI2);
      Indice2 := ((DSR.dsDays - LI1) MOD LI2);
      i := SHORT (Indice1);
      CASE i OF
        0 : Ligne := "       Lundi"|
        1 : Ligne := "       Mardi"|
        2 : Ligne := "       Mercredi"|
        3 : Ligne := "       Jeudi"|
        4 : Ligne := "       Vendredi"|
        5 : Ligne := "       Samedi"|
        6 : Ligne := "       Dimanche"
      END;
      FiNa := "Param:";
      ConcatString (FiNa,Nom);
      ConcatString (FiNa,St1);
      ConcatString (FiNa,".RESUME");
      OpenOutputFile (FiNa);
      WriteLn;
      WriteString ("     ");
      WriteString (Nom);
      WriteString ("     ");
      WriteString (Ligne);
      WriteLn;
      WriteLn;
      WriteString ("     SOLDES DES COMPTES EN DEBUT DE SESSION : ");
      ConvRealToString (TRecettes.Tab[0].Valeur,Montant,2,Decimal);
      WriteString (Montant);
      WriteLn;
      WriteLn;
      WriteString ("     RECETTES ENREGISTREES");
      WriteLn;
      WriteLn;
      WriteString ("     MOMENT              POSTE                    MONTANT "
      WriteLn;
      WriteLn;
      imp := 1;
      WHILE imp <= TRecettes.Indice-1 DO
        Ligne := "                                                          ";
        OverwriteWithSubString (Ligne,TRecettes.Tab[imp].Jour,5);
        OverwriteWithSubString (Ligne,TRecettes.Tab[imp].Poste,24);
        ConvRealToString (TRecettes.Tab[imp].Valeur,Montant,2,Decimal);
        OverwriteWithSubString (Ligne,Montant,53-StringLength(Montant));
        WriteString (Ligne);
        WriteLn;
        imp := imp + 1;
      END;
      WriteLn;
      WriteString ("     DEPENSES ENREGISTREES");
      WriteLn;
      WriteLn;
      WriteString ("     MOMENT              POSTE                    MONTANT
```

```
      WriteLn;
      WriteLn;
      imp := 0;
      WHILE imp <= TDepenses.Indice-1 DO
        Ligne := "                                                                    ";
        OverwriteWithSubString (Ligne,TDepenses.Tab[imp].Jour,5);
        OverwriteWithSubString (Ligne,TDepenses.Tab[imp].Poste,24);
        ConvRealToString (TDepenses.Tab[imp].Valeur,Montant,2,Decimal);
        OverwriteWithSubString (Ligne,Montant,53-StringLength(Montant));
        WriteString (Ligne);
        WriteLn;
        imp := imp + 1;
      END;
      WriteLn;
      WriteString ("     SOLDES DES COMPTES EN FIN DE SESSION : ");
      ConvRealToString (Sactuel,Montant,2,Decimal);
      WriteString (Montant);
      WriteLn;
      WriteLn;
      WriteString ("     ETAT DU PORTE-MONNAIE : ");
      ConvRealToString (Sinit,Montant,2,Decimal);
      WriteString (Montant);
      WriteLn;
      CloseOutput();
END ImpApres;

 PROCEDURE SaisieFonction (ValeurFonction : Str20);
    BEGIN
      IF (CompareString (ValeurFonction,"recettes") = equal)
          THEN EnregistrerRecette;
      ELSIF (CompareString (ValeurFonction,"depenses") = equal)
          THEN EnregistrerDepense;
      ELSIF (CompareString (ValeurFonction,"portemonnaie") = equal)
          THEN EtatPorteMonnaie.EPM;
      ELSIF (CompareString (ValeurFonction,"correction")=equal)
          THEN CorrectionPM END;
    END SaisieFonction;



 PROCEDURE Initialisation;
    VAR j : INTEGER;
        p : Str20;


    BEGIN
      (* Garniture du tableau des parametres *)

      FileName := "Param:";
      ConcatString (FileName,"UTILISATEUR");
      ConcatString (FileName,".PARAM");
      OpenInputFile (FileName);
      ScreenToFront (Scr^);
      IF NOT Done THEN
                      TableauParametres[0] := 0;
                      TableauParametres[1] := 2;
                      TableauParametres[2] := 0;
                      TableauParametres[3] := 1;
                      TableauParametres[4] := 6;
                      TableauParametres[5] := 0;
                      TableauParametres[6] := 0;
                      TableauParametres[7] := 0;
                      TableauParametres[8] := 0;
                      TableauParametres[9] := 1;
                      TableauParametres[10] := 0;
                      TableauParametres[11] := 1;
```

```
                        TableauParametres[12] := 2;
                    ELSE
                        j := 0;
                        ReadString (p);
                        WHILE (j <= 12) DO

                           CASE p[j] OF
                             "0" : TableauParametres[j] := 0 |
                             "1" : TableauParametres[j] := 1 |
                             "2" : TableauParametres[j] := 2 |
                             "3" : TableauParametres[j] := 3 |
                             "4" : TableauParametres[j] := 4 |
                             "5" : TableauParametres[j] := 5 |
                             "6" : TableauParametres[j] := 6
                           END;
                           j := j + 1;
                           END;
                        CloseInput;
                        END;

        TableauTraces.Tab[TableauTraces.Indice] := " ";
        TableauTraces.Indice := TableauTraces.Indice + 1;
        TableauTraces.Tab[TableauTraces.Indice] := "Liste des paramètres choisi
";
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[0] = 0
          THEN TableauTraces.Tab[TableauTraces.Indice] := "Pas d'impression"
          ELSE TableauTraces.Tab[TableauTraces.Indice] := "Impression du résumé
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[1] = 0
          THEN TableauTraces.Tab[TableauTraces.Indice] := "Questions posées par
écrit"
          ELSE TableauTraces.Tab[TableauTraces.Indice] := "Questions posées par
écrit + parole"
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[2] = 0
          THEN TableauTraces.Tab[TableauTraces.Indice] := "Saisie du Nom par c
vier"
          ELSE TableauTraces.Tab[TableauTraces.Indice] := "Saisie du Nom par s
ris"
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[3] = 0
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Recettes saisies par
avier";
        END;
        IF TableauParametres[3] = 1
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Recettes saisies par
llets";
        END;
        IF TableauParametres[3] = 2
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Recettes saisies par
lculette";
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[12] = 0
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Dépenses saisies par
avier";
        END;
        IF TableauParametres[12] = 1
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Dépenses saisies par
llets";
        END;
        IF TableauParametres[12] = 2
```

```
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Dépenses saisies par c
lculette";
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        CASE TableauParametres[4] OF
          0: TableauTraces.Tab[TableauTraces.Indice] := "Porte-Monnaie -> nombr
s"|
          1: TableauTraces.Tab[TableauTraces.Indice] := "Porte-Monnaie -> dessi
s"|
          3: TableauTraces.Tab[TableauTraces.Indice] := "Porte-Monnaie -> nombr
s + dessins"|
          4: TableauTraces.Tab[TableauTraces.Indice] := "Porte-Monnaie -> nombr
s + parole"|
          5: TableauTraces.Tab[TableauTraces.Indice] := "Porte-Monnaie -> dessi
s + parole"|
          6: TableauTraces.Tab[TableauTraces.Indice] := "Porte-Monnaie -> nbres-
dessins+parole"
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[11] = 0
          THEN TableauTraces.Tab[TableauTraces.Indice] := "Réglettes absentes"
          ELSE TableauTraces.Tab[TableauTraces.Indice] := "Réglettes présentes"
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[6] = 0
          THEN TableauTraces.Tab[TableauTraces.Indice] := "Notion du temps prés
nte"
          ELSE TableauTraces.Tab[TableauTraces.Indice] := "Notion du temps abse
te"
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[7] = 0
          THEN TableauTraces.Tab[TableauTraces.Indice] := "Poste de dépense à s
écifier"
          ELSE TableauTraces.Tab[TableauTraces.Indice] := "Pas de poste de dépe
se à spécifier"
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[8] = 0
          THEN TableauTraces.Tab[TableauTraces.Indice] := "Récapitulatif préser
"
          ELSE TableauTraces.Tab[TableauTraces.Indice] := "Récapitulatif absent
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[9] = 0
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Cycle d'un jour"
        END;
        IF TableauParametres[9] = 2
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Cycle d'un mois"
        END;
        IF TableauParametres[9] = 1
        THEN TableauTraces.Tab[TableauTraces.Indice] := "Cycle d'une semaine"
        END;
        TableauTraces.Indice := TableauTraces.Indice + 1;
        IF TableauParametres[9] = 1
          THEN
            CASE TableauParametres[10] OF
              0: TableauTraces.Tab[TableauTraces.Indice] := "Début de cycle =
ndi"|
              1: TableauTraces.Tab[TableauTraces.Indice] := "Début de cycle =
rdi"|
              2: TableauTraces.Tab[TableauTraces.Indice] := "Début de cycle =
rcredi"|
              3: TableauTraces.Tab[TableauTraces.Indice] := "Début de cycle =
udi"|
              4: TableauTraces.Tab[TableauTraces.Indice] := "Début de cycle =
```

```
ndredi";
            5: TableauTraces.Tab[TableauTraces.Indice] := "Début de cycle = s
medi";
            6: TableauTraces.Tab[TableauTraces.Indice] := "Début de cycle = d
manche"
          END;
          TableauTraces.Indice := TableauTraces.Indice + 1;
          TableauTraces.Tab[TableauTraces.Indice] := " ";
          TableauTraces.Indice := TableauTraces.Indice + 1;
        ELSE
          TableauTraces.Tab[TableauTraces.Indice] := " ";
          TableauTraces.Indice := TableauTraces.Indice + 1;
      END;

      (* Garniture de Max *)

      FileName := "Param:";
      ConcatString (FileName,Nom);
      ConcatString (FileName,".MAX");
      OpenInputFile (FileName);
      ScreenToFront (Scr^);
      IF NOT Done
            THEN
                  Max := 300.
            ELSE
                  ReadReal (Max);
                  CloseInput;
      END;

      (* Garniture de Sactuel *)

      FileName := "Param:";
      ConcatString (FileName,Nom);
      ConcatString (FileName,".SOMME");
      OpenInputFile (FileName);
      ScreenToFront (Scr^);
      IF NOT Done THEN Sactuel := 0. ;
                  ELSE
                        ReadReal (s);
                        Sactuel := s;
                        CloseInput;
                        END;

      (* Initialisation des tableaux TRecettes et TDepenses *)
      TRecettes.Tab[0].Valeur := Sactuel;
      TRecettes.Indice := 1;
      TDepenses.Indice := 0;
      Sinit :=0.;

      (* Initialisation du tableau des nombres de billets et pieces
            eventuellement enregistres par l'ecran graphique *)

      j:= 0;
      WHILE j <= 15 DO
         TabNombre[j] := 0;
         j := j + 1;
      END;

    TempsSommeInit:=0;
    TempsSIBillets:=0;
    TempsCorrection:=0;
    TempsSommeRec:=0;
    TempsMomentRec:=0;
    TempsPosteRec:=0;
    TempsSommeDep:=0;
    TempsMomentDep:=0;
```

```
        TempsPosteDep:=0;
        TempsTotal :=0;

        DateStamp(DSR);
        MinTotBEG := SHORT (DSR.dsMinute);
        TicksTotBEG := SHORT (DSR.dsTick);
        IF TableauParametres[11] = 1
        THEN Reglette ("R",TRUE);
        END;

      END Initialisation;

PROCEDURE InitPalette;

BEGIN

        (* Initialisation du tableau cmap des 32 couleurs de reference *)
          cmap[00] := 0000H;
          cmap[01] := 0FFFH;
          cmap[02] := 0E00H;
          cmap[03] := 0A00H;
          cmap[04] := 0D80H;
          cmap[05] := 0FE0H;
          cmap[06] := 08F0H;
          cmap[07] := 0080H;
          cmap[08] := 00B6H;
          cmap[09] := 00DDH;
          cmap[10] := 00AFH;
          cmap[11] := 007CH;
          cmap[12] := 000FH;
          cmap[13] := 070FH;
          cmap[14] := 0C0EH;
          cmap[15] := 0C08H;
          cmap[16] := 0620H;
          cmap[17] := 0E52H;
          cmap[18] := 0A52H;
          cmap[19] := 0FCAH;
          cmap[20] := 0333H;
          cmap[21] := 0444H;
          cmap[22] := 0555H;
          cmap[23] := 0666H;
          cmap[24] := 0777H;
          cmap[25] := 0888H;
          cmap[26] := 0999H;
          cmap[27] := 0AAAH;
          cmap[28] := 0CCCH;
          cmap[29] := 0DDDH;
          cmap[30] := 0EEEH;
          cmap[31] := 0FFFH;
          LoadRGB4 (Scr^.ViewPort,ADR (cmap),2);
END InitPalette;


   PROCEDURE Terminaison;
      VAR i : INTEGER;

      BEGIN

        DateStamp(DSR);
        MinEND := SHORT(DSR.dsMinute);
        TicksEND := SHORT (DSR.dsTick);
        IF MinEND > MinTotBEG
        THEN TicksEND := TicksEND + (3000 * (MinEND - MinTotBEG))
        END;
        ent := TicksEND - TicksTotBEG;
        ree := real(ent);
```

```
TempsTotal := entier (ree / 50.);

(* Sauvetage somme restante *)

FileName := "Param:";
ConcatString (FileName,Nom);
ConcatString (FileName,".SOMME");
OpenOutputFile (FileName);
WriteReal (Sinit,8);
CloseOutput;

(* Sauvetage trace *)

FileName := "Param:";
ConcatString (FileName,Nom);
CopyString (FileName2,FileName);
ConcatString (FileName2,".DV");
OpenInputFile (FileName2);
IF Done THEN ReadReal (DernVers);
             DernVers := DernVers + 1.;
        ELSE DernVers := 0.
END;
CloseInput;

OpenOutputFile (FileName2);
WriteReal (DernVers,4);
CloseOutput;
ConvRealToString (DernVers,St1,0,Decimal);
ConcatString (FileName,St1);
CopyString (FileName3,FileName);
ConcatString (FileName3,".TEMPS");

OpenOutputFile (FileName3);
WriteInt(TempsSommeInit,4);
WriteLn;
WriteInt(TempsSIBillets,4);
WriteLn;
WriteInt(TempsCorrection,4);
WriteLn;
WriteInt(TempsSommeRec,4);
WriteLn;
WriteInt(TempsMomentRec,4);
WriteLn;
WriteInt(TempsPosteRec,4);
WriteLn;
WriteInt(TempsSommeDep,4);
WriteLn;
WriteInt(TempsMomentDep,4);
WriteLn;
WriteInt(TempsPosteDep,4);
WriteLn;
WriteInt(TempsTotal,4);
WriteLn;
CloseOutput;

ConcatString (FileName,".TRACE");
CopyString (Ph,Nom);
ConcatString (Ph," : trace n° ");
ConcatString (Ph,St1);
CopyString (TableauTraces.Tab[0],Ph);

IF (Sinit = Sactuel)
     THEN  Ph := "Les comptes correspondent à la réalité";
     ELSIF (Sinit > Sactuel)
        THEN Ph := "Comptes < réalité : manque recettes";
        ELSE Ph := "Comptes > réalité : manque dépenses";
```

```
      END;
      IF TableauTraces.Indice <= 499 THEN
                  CopyString (TableauTraces.Tab[TableauTraces.Indice],Ph);
                  TableauTraces.Indice := TableauTraces.Indice + 1;
      END;
      OpenOutputFile (FileName);
      i := 0;
      WHILE (i <= TableauTraces.Indice) DO
         WriteString (TableauTraces.Tab[i]);
         WriteLn;
         i := i + 1;
         END;
      CloseOutput;

      (* Sauvetage chaine *)

      FileName := "Param:";
      ConcatString (FileName,Nom);
      ConcatString (FileName,St1);
      ConcatString (FileName,".CHAINE");
      OpenOutputFile(FileName);
      i := 0;
      WHILE (i < TableauChaine.Indice) DO
         WriteInt(TableauChaine.Tab[i],1);
         WriteLn;
         i := i + 1;
      END;
      CloseOutput;

      FileName := "Param:";
      ConcatString (FileName,Nom);
      ConcatString (FileName,St1);
      ConcatString (FileName,".LONG_CHAINE");
      OpenOutputFile(FileName);
      WriteInt(TableauChaine.Indice,1);
      CloseOutput;

      (* Sauvetage des temps *)

      FileName := "Param:";
      ConcatString (FileName,Nom);
      ConcatString (FileName,St1);
      ConcatString (FileName,".TPS");
      OpenOutputFile(FileName);
      i := 0;
      WHILE (i < TableauChaine.Indice) DO
         WriteInt(TableauTemps[i],4);
         WriteLn;
         i := i + 1;
      END;
      CloseOutput;

      ImpApres;
      IF TableauParametres[0] = 1 THEN Impression END;
    END Terminaison;

BEGIN (* coordinateur *)
  Scr := CreateScreen (320,256,1,NIL);
  IF Scr # NIL THEN
  InitPalette;
  Win := CreateWindow (0,0,320,256,WIDCMP,WFlags,NIL,Scr,NIL);
  IF (Win # NIL) THEN
  TableauTraces.Indice := 1;
  Presentation;
  SSaisieNom;
  Initialisation;
```

```
  EnregistrerSommeInitiale (EtatSortie);
  Q := FALSE; FinProg := FALSE;
  IF (CompareString (EtatSortie,"quitter") # equal)
        THEN
              WHILE NOT Q DO
                 Menu (Fonction);
                 WHILE (CompareString (Fonction,"fin") # equal) DO
                   SaisieFonction (Fonction);
                   Menu (Fonction);
                 END;
                 IF TableauParametres[11] = 1 THEN
                 Reglette ("T",FALSE);
                 END;
                 Equilibre (Q);
              END;
         END;
  Terminaison;
  CloseWindow (Win^);
  END;
  CloseScreen (Scr^);
  END;
END Coordinateur.
```

```
IMPLEMENTATION MODULE EtatPorteMonnaie;

FROM VariablesGlobales IMPORT TRecettes, TDepenses, Sinit, Sactuel,
                              TableauParametres,TableauTraces,TableauChaine,
                              AncMinute,AncTick,TableauTemps ;
FROM MathLib0 IMPORT entier, real;
FROM SYSTEM IMPORT SHORT;
FROM AmigaDos IMPORT DateStamp,DateStampRecord;
FROM ModEquilibre IMPORT Equilibre;
FROM Interface IMPORT PorteMonnaie, EPMRecette, EPMDepense, PresentationRecDep;
FROM RealConversions IMPORT ConvRealToString,RealToStringFormat;
FROM Strings IMPORT CompareString,Relation,CopyString;
FROM InOut IMPORT WriteString,WriteLn;
FROM ModReglette IMPORT Reglette;

VAR  ActMinute,ActTick,ent,ActTick2 : INTEGER;
     ree : REAL;
     DSR : DateStampRecord;

PROCEDURE EPM;
  VAR SI,SA : ARRAY [0..19] OF CHAR;
      phrase : ARRAY [0..40] OF CHAR;

  BEGIN

      IF TableauChaine.Indice <= 499
      THEN TableauChaine.Tab[TableauChaine.Indice] :=24;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
           TableauChaine.Indice := TableauChaine.Indice + 1;
           TableauChaine.Tab[TableauChaine.Indice] :=7;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
           TableauChaine.Indice := TableauChaine.Indice + 1;
      END;

       IF (Sinit = Sactuel)
            THEN PorteMonnaie;
                 IF TableauParametres[11] = 1
                 THEN Reglette ("S",TRUE);
                 END;
                 IF TableauParametres[8] = 0 THEN
                 PresentationRecDep;
                 IF TableauParametres[11] = 1
                 THEN Reglette ("T",FALSE);
                 END;
```

```
                    phrase := 'On presente le recapitulatif Rec-Dep';
                     IF TableauTraces.Indice <= 499 THEN
                      CopyString (TableauTraces.Tab[TableauTraces.Indice],phrase)
                      TableauTraces.Indice := TableauTraces.Indice + 1;
                     END;
                    END;
                ELSIF (Sinit > Sactuel)
                    THEN
                     EPMRecette;
                     IF TableauParametres[11] = 1
                     THEN Reglette ("S",TRUE);
                     END;
                     IF TableauParametres[8] = 0 THEN
                          PresentationRecDep;
                          IF TableauParametres[11] = 1
                          THEN Reglette ("T",FALSE);
                          END;
                          phrase := "On presente le recapitulatif Rec-Dep";
                           IF TableauTraces.Indice <= 499 THEN
                            CopyString (TableauTraces.Tab[TableauTraces.Indice],p
rase);
                             TableauTraces.Indice := TableauTraces.Indice + 1;
                            END;
                     END;
                    ELSE
                     EPMDepense;
                     IF TableauParametres[11] = 1
                     THEN Reglette ("S",TRUE);
                     END;
                     IF TableauParametres[8] = 0 THEN
                     PresentationRecDep;
                     IF TableauParametres[11] = 1
                     THEN Reglette ("T",FALSE);
                     END;
                     phrase := "On presente le recapitulatif Rec-Dep";
                          IF TableauTraces.Indice <= 499 THEN
                          CopyString (TableauTraces.Tab[TableauTraces.Indice],phr
se);
                          TableauTraces.Indice := TableauTraces.Indice + 1;
                          END;
                      END;
                   END;

  END EPM;

END EtatPorteMonnaie.
```

```
IMPLEMENTATION MODULE EnregistrerSomme;
  FROM VariablesGlobales IMPORT TableauParametres, TDepenses, TRecettes,
                                Sinit, Sactuel, Str20, TableauChaine,
                                AncMinute,AncTick,TableauTemps ;
  FROM SYSTEM IMPORT SHORT;
  FROM ModReglette IMPORT Reglette;
  FROM MathLib0 IMPORT entier,real;
  FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
  FROM Interface IMPORT Recettes, Depenses, SommeInitiale, MomentDepense,
                        PosteDepense, PosteRecette;
  FROM Strings IMPORT Relation,CompareString;
  VAR EtatSortie : Str20;
      ActMinute,ActTick,ent,ActTick2 : INTEGER;
      ree : REAL;
      DSR : DateStampRecord;

  PROCEDURE EnregistrerRecette;
     VAR ValeurRecette : REAL;
         ValeurJour, ValeurPoste : Str20;
         Mouv : CHAR;
     BEGIN
       IF TableauParametres[7] = 0
       THEN IF TableauChaine.Indice <= 499
            THEN TableauChaine.Tab [TableauChaine.Indice] := 8 ;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            PosteRecette (ValeurPoste)
       END;
       IF TableauParametres[6] = 0
       THEN Mouv := "R";
            IF TableauChaine.Indice <= 499
            THEN TableauChaine.Tab [TableauChaine.Indice] := 10;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            MomentDepense (ValeurJour,Mouv)
       END;
       IF TableauChaine.Indice <= 499
       THEN TableauChaine.Tab [TableauChaine.Indice] := 12;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
```

```
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
          TableauChaine.Indice := TableauChaine.Indice + 1;
     END;
     Recettes (ValeurRecette, EtatSortie);
     IF CompareString (EtatSortie,"quitter") <> equal THEN
          Sactuel := Sactuel + ValeurRecette;
          TRecettes.Tab[TRecettes.Indice].Valeur := ValeurRecette;
          IF TableauParametres[6] = 0 THEN
               TRecettes.Tab[TRecettes.Indice].Jour   := ValeurJour END;
          IF TableauParametres[7] = 0 THEN
               TRecettes.Tab[TRecettes.Indice].Poste  := ValeurPoste END;
          TRecettes.Indice := TRecettes.Indice + 1;
          IF TableauParametres[11] = 1 THEN
               Reglette ("R",TRUE);   END;
     END;
   END EnregistrerRecette;

PROCEDURE EnregistrerDepense;
   VAR ValeurDepense : REAL;
       ValeurJour, ValeurPoste : Str20;
       Mouv : CHAR;
   BEGIN
     IF TableauParametres[7] = 0
     THEN IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab [TableauChaine.Indice] := 16;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
          PosteDepense (ValeurPoste)
     END;
     IF TableauParametres[6] = 0
     THEN Mouv := "D";
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab [TableauChaine.Indice] := 18;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
```

```
            MomentDepense (ValeurJour,Mouv)
        END;
        IF TableauChaine.Indice <= 499
        THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
        END;
        Depenses (ValeurDepense, EtatSortie);
        IF CompareString (EtatSortie,"quitter") <> equal THEN
            Sactuel := Sactuel - ValeurDepense;
            TDepenses.Tab[TDepenses.Indice].Valeur := ValeurDepense;
            IF TableauParametres[6] = 0 THEN
                TDepenses.Tab[TDepenses.Indice].Jour   := ValeurJour END;
            IF TableauParametres[7] = 0 THEN
                TDepenses.Tab[TDepenses.Indice].Poste  := ValeurPoste END;
            TDepenses.Indice := TDepenses.Indice + 1;
            IF TableauParametres[11] = 1 THEN
                Reglette ("D",TRUE);  END;
        END;
    END EnregistrerDepense;

  PROCEDURE EnregistrerSommeInitiale (VAR EtatSortie2 : Str20);
    VAR ValeurSommeInitiale : REAL;
    BEGIN
      IF TableauChaine.Indice <= 499
      THEN TableauChaine.Tab [TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
      END;
      SommeInitiale (ValeurSommeInitiale, EtatSortie);
      IF CompareString (EtatSortie,"quitter") <> equal
                THEN Sinit := ValeurSommeInitiale;
                ELSE EtatSortie2 := "quitter" END;
      IF TableauParametres[11] = 1
      THEN Reglette ("S",TRUE);
      END;
    END EnregistrerSommeInitiale;

END EnregistrerSomme.
```

```
IMPLEMENTATION MODULE Interface;

FROM SYSTEM IMPORT ADR,SHORT;
FROM AmigaDOS IMPORT DeleteFile, DateStamp, DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM Strings IMPORT Relation, CompareString, ConvStringToUpperCase, CopyString,
                    ConcatString , StringLength;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat;
FROM VariablesGlobales IMPORT TableauParametres, TableauTraces, Nom, Str20,
                              Sinit;
FROM VariablesGlobales IMPORT TempsSommeInit, TempsSommeRec, TempsMomentRec,
      TempsPosteRec,TempsSommeDep,TempsMomentDep,TempsPosteDep,TempsCorrection;
FROM InOut IMPORT OpenInputFile,CloseInput,OpenOutputFile,CloseOutput,
                  ReadString,Done;
FROM Intuition IMPORT ScreenToFront,ScreenPtr,CloseScreen;
FROM SimpleScreens IMPORT CreateScreen;
IMPORT ModMenu;
IMPORT ModNom;
IMPORT ModNomSou;
FROM ModMoment IMPORT Moment;
FROM ModJour IMPORT Jour;
FROM ModMois IMPORT Mois;
FROM Scalcu IMPORT SaisieSommeCalcu;
FROM ModPoste IMPORT Poste;
FROM ModPosteRec IMPORT PosteRec;
IMPORT PRD;
FROM SRBillets IMPORT SommeRestanteBillets;
FROM SRNombre IMPORT SommeRestanteNombre;
FROM SSBillets IMPORT SaisieSommeBillets;
FROM SSNombre IMPORT SaisieSommeNbre;
FROM CorrPMBillets IMPORT CorrPMBI;
FROM CorrectionPMCL IMPORT CorrPMCL;
FROM CorrectionPMCA IMPORT CorrPMCA;
FROM ModReglette IMPORT Reglette;

VAR EtatSortie,p : Str20;
    DSR : DateStampRecord;
    NminBEG, NminEND, NticksBEG, NticksEND : INTEGER;
    ent : INTEGER;
    ree : REAL;
    j : INTEGER;
    FileName : ARRAY [0..40] OF CHAR;
    Scr : ScreenPtr;

(*****************************************************************************
(*                              SaisieNom                                    *
(*****************************************************************************

PROCEDURE SSaisieNom;

  PROCEDURE EnleveBlancs (VAR S : Str20);
  VAR Indice1,Indice2 : CARDINAL;
      S2 : Str20;
  BEGIN
    Indice1 := 0;
    Indice2 := 0;
    WHILE (Indice1 <= (StringLength (S) - 1)) DO
      IF (S[Indice1] # " ") AND (S[Indice1] # "-") AND (S[Indice1] # "_")
      THEN
        IF (S[Indice1] = "é") OR (S[Indice1] = "è")
        THEN S2[Indice2] := "E";
             Indice2 := Indice2 + 1;
        ELSE
           S2[Indice2] := S[Indice1];
           Indice2 := Indice2 + 1;
        END;
```

```
      END (* IF *);
      Indice1 := Indice1 + 1;
    END (* WHILE *);
    S2[Indice2] := S[Indice1];
    CopyString (S,S2);
  END EnleveBlancs;

BEGIN (* SSaisieNom *)

        Scr := CreateScreen(320,200,5,NIL);
        FileName := "Param:";
        ConcatString (FileName,"UTILISATEUR");
        ConcatString (FileName,".PARAM");
        OpenInputFile (FileName);
        ScreenToFront (Scr^);
        IF NOT Done THEN
                        TableauParametres[0] := 0;
                        TableauParametres[1] := 2;
                        TableauParametres[2] := 0;
                        TableauParametres[3] := 1;
                        TableauParametres[4] := 6;
                        TableauParametres[5] := 0;
                        TableauParametres[6] := 0;
                        TableauParametres[7] := 0;
                        TableauParametres[8] := 0;
                        TableauParametres[9] := 1;
                        TableauParametres[10] := 0;
                        TableauParametres[11] := 1;
                        TableauParametres[12] := 2;
                    ELSE
                        j := 0;
                        ReadString (p);
                        WHILE (j <= 12) DO

                           CASE p[j] OF
                             "0" : TableauParametres[j] := 0 ¦
                             "1" : TableauParametres[j] := 1 ¦
                             "2" : TableauParametres[j] := 2 ¦
                             "3" : TableauParametres[j] := 3 ¦
                             "4" : TableauParametres[j] := 4 ¦
                             "5" : TableauParametres[j] := 5 ¦
                             "6" : TableauParametres[j] := 6
                           END;
                           j := j + 1;
                           END;
                        CloseInput;
                    END;
  CloseScreen(Scr^);
  IF TableauParametres[2] = 0
  THEN ModNom.SaisieNom;
  ELSE
    ModNomSou.SaisieNomSou;
  END;
  EnleveBlancs (Nom);
  ConvStringToUpperCase (Nom);
END SSaisieNom;


(*****************************************************************************
(*                              Menu
(*****************************************************************************

PROCEDURE Menu (VAR FonctionChoisie : Str20);

  VAR phrase : ARRAY [0..40] OF CHAR;
  BEGIN
```

```
        ModMenu.Menu (FonctionChoisie);

        (* trace du choix *)
        CopyString (phrase,"Il choisit la fonction : ");
        ConcatString (phrase,FonctionChoisie);
        ConcatString (phrase," ");
        IF TableauTraces.Indice <= 499 THEN
        CopyString (TableauTraces.Tab[TableauTraces.Indice],phrase);
        TableauTraces.Indice := TableauTraces.Indice + 1;
        END;
  END Menu;


(*****************************************************************************
(*                                Recettes                                   *
(*****************************************************************************
PROCEDURE Recettes (VAR ValeurRecette : REAL ; VAR EtatSortie1 : Str20);
VAR Phrase : ARRAY[0..40] OF CHAR;
    PhrasePron : ARRAY[0..99] OF CHAR;
    Trace : ARRAY[0..40] OF CHAR;
    SuiteTrace : ARRAY[0..10] OF CHAR;
BEGIN
  Phrase := "Combien avez-vous reçu d'argent ?";
  PhrasePron := "Cobya ah vay voo ra su darjo ?";
  DateStamp(DSR);
  NminBEG := SHORT (DSR.dsMinute);
  NticksBEG := SHORT (DSR.dsTick);
  IF (TableauParametres[3] = 0)
  THEN SaisieSommeNbre (Phrase,PhrasePron,TRUE,ValeurRecette,
                                        EtatSortie1);
  END;
  IF (TableauParametres[3] = 1)
  THEN SaisieSommeBillets (Phrase,PhrasePron,TRUE,ValeurRecette,
                                        EtatSortie1);
  END;
  IF (TableauParametres[3] = 2)
  THEN SaisieSommeCalcu (Phrase,PhrasePron,TRUE,ValeurRecette,
                                        EtatSortie1);
  END (* IF *);
  DateStamp(DSR);
  NminEND := SHORT (DSR.dsMinute);
  NticksEND := SHORT (DSR.dsTick);
        IF TempsSommeRec = 0
        THEN
          IF NminEND > NminBEG
          THEN NticksEND := NticksEND + (3000 * (NminEND - NminBEG))
          END;
          ent := NticksEND - NticksBEG;
          ree := real(ent);
          TempsSommeRec := entier (ree / 50.)
        END;
  (* Trace de la recette *)
  IF (CompareString (EtatSortie1,"quitter") # equal) THEN
   CopyString (Trace, "Il donne une recette de : ");
   ConvRealToString (ValeurRecette,SuiteTrace,2,Decimal);
   ConcatString (Trace,SuiteTrace);
   ConcatString (Trace," frs ");
       IF TableauTraces.Indice <= 499 THEN
   CopyString (TableauTraces.Tab[TableauTraces.Indice],Trace);
   TableauTraces.Indice := TableauTraces.Indice + 1;
        END;
  END (* IF *);
END Recettes;
(*****************************************************************************
(*                                Depenses
(*****************************************************************************
```

```
PROCEDURE Depenses (VAR ValeurDepense : REAL ; VAR EtatSortie1 : Str20);
VAR Phrase : ARRAY[0..40] OF CHAR;
    PhrasePron : ARRAY[0..99] OF CHAR;
    Trace : ARRAY[0..40] OF CHAR;
    SuiteTrace : ARRAY[0..10] OF CHAR;
BEGIN
  Phrase := "Combien avez-vous dépensé ?";
  PhrasePron := "Cobya ah vay voo daypensay ?";
  DateStamp(DSR);
  NminBEG := SHORT (DSR.dsMinute);
  NticksBEG := SHORT (DSR.dsTick);
  IF (TableauParametres[12] = 0)
  THEN SaisieSommeNbre (Phrase,PhrasePron,TRUE,ValeurDepense,
                                        EtatSortie1);
  END;
  IF (TableauParametres[12] = 1)
  THEN SaisieSommeBillets (Phrase,PhrasePron,TRUE,ValeurDepense,
                                        EtatSortie1);
  END;
  IF (TableauParametres[12] = 2)
  THEN SaisieSommeCalcu (Phrase,PhrasePron,TRUE,ValeurDepense,
                                        EtatSortie1);
  END (* IF *);
  DateStamp(DSR);
  NminEND := SHORT (DSR.dsMinute);
  NticksEND := SHORT (DSR.dsTick);

        IF TempsSommeDep = 0
        THEN
          IF NminEND > NminBEG
          THEN NticksEND := NticksEND + (3000 * (NminEND - NminBEG))
          END;
          ent := NticksEND - NticksBEG;
          ree := real (ent);
          TempsSommeDep := entier (ree / 50.)
        END;
  (* Trace de la dépense *)
  IF (CompareString (EtatSortie1,"quitter") # equal) THEN
   CopyString (Trace, "Il donne une dépense de : ");
   ConvRealToString (ValeurDepense,SuiteTrace,2,Decimal);
   ConcatString (Trace,SuiteTrace);
   ConcatString (Trace," frs ");
       IF TableauTraces.Indice <= 499 THEN
   CopyString (TableauTraces.Tab[TableauTraces.Indice],Trace);
   TableauTraces.Indice := TableauTraces.Indice + 1;
       END;
  END (* IF *);
END Depenses;
(*****************************************************************************)
(*                              SommeInitiale                                *)
(*****************************************************************************)
PROCEDURE SommeInitiale (VAR ValeurSInitiale : REAL ; VAR EtatSortie1 : Str20);

VAR Phrase : ARRAY[0..40] OF CHAR;
    PhrasePron : ARRAY[0..99] OF CHAR;
    Trace : ARRAY[0..40] OF CHAR;
    SuiteTrace : ARRAY[0..10] OF CHAR;
BEGIN
  Phrase := "Combien avez-vous d'argent ?";
  PhrasePron := "Cobya ah vay voo  darjo ?";
  DateStamp(DSR);
  NminBEG := SHORT (DSR.dsMinute);
  NticksBEG := SHORT (DSR.dsTick);
  IF (TableauParametres[3] = 0)
  THEN SaisieSommeNbre (Phrase,PhrasePron,FALSE,ValeurSInitiale,
                                        EtatSortie1);
```

```
  END;
  IF (TableauParametres[3] = 1)
  THEN SaisieSommeBillets (Phrase,PhrasePron,FALSE,ValeurSInitiale,
                                          EtatSortie1);
  END;
  IF (TableauParametres[3] = 2)
  THEN SaisieSommeCalcu (Phrase,PhrasePron,FALSE,ValeurSInitiale,
                                          EtatSortie1);
  END (* IF *);
    DateStamp(DSR);
  NminEND := SHORT (DSR.dsMinute);
  NticksEND := SHORT (DSR.dsTick);

        IF TempsSommeInit = 0
        THEN
          IF NminEND > NminBEG
          THEN NticksEND := NticksEND + (3000 * (NminEND - NminBEG))
          END;
          ent := NticksEND - NticksBEG;
          ree := real(ent);
          TempsSommeInit := entier (ree / 50.)
        END;
  (* Trace de la somme initiale *)
  IF (CompareString (EtatSortie1,"quitter") # equal) THEN
   CopyString (Trace, "Il donne une s. initiale de : ");
   ConvRealToString (ValeurSInitiale,SuiteTrace,2,Decimal);
   ConcatString (Trace,SuiteTrace);
   ConcatString (Trace," frs ");
       IF TableauTraces.Indice <= 499 THEN
   CopyString (TableauTraces.Tab[TableauTraces.Indice],Trace);
   TableauTraces.Indice := TableauTraces.Indice + 1;
        END;
  END (* IF *);
END SommeInitiale;

(*****************************************************************************)
(*                          MomentDepense                                    *)
(*****************************************************************************)

PROCEDURE MomentDepense (VAR ValeurMoment : Str20; Mouvement : CHAR);


   VAR phrase : ARRAY[0..40] OF CHAR;

   BEGIN
     DateStamp(DSR);
     NminBEG := SHORT (DSR.dsMinute);
     NticksBEG := SHORT (DSR.dsTick);
     IF (TableauParametres[9] = 0)
     THEN
        Moment (ValeurMoment,Mouvement);
        phrase := "Il donne le moment : ";
        ConcatString (phrase,ValeurMoment);
        IF TableauTraces.Indice <= 499 THEN
        CopyString (TableauTraces.Tab[TableauTraces.Indice],phrase);
        TableauTraces.Indice := TableauTraces.Indice + 1;
        END;
     END;
     IF (TableauParametres[9] = 1)
     THEN
        Jour (TableauParametres[10],ValeurMoment,Mouvement);
        phrase := "Il donne le jour : ";
        ConcatString (phrase,ValeurMoment);
        IF TableauTraces.Indice <= 499 THEN
        CopyString (TableauTraces.Tab[TableauTraces.Indice],phrase);
        TableauTraces.Indice := TableauTraces.Indice + 1;
```

```
      END;
   END;
   IF (TableauParametres[9] = 2)
   THEN
      Mois (ValeurMoment,Mouvement);
      phrase := "Il donne le jour : ";
      ConcatString (phrase,ValeurMoment);
      IF TableauTraces.Indice <= 499 THEN
      CopyString (TableauTraces.Tab[TableauTraces.Indice],phrase);
      TableauTraces.Indice := TableauTraces.Indice + 1;
      END;
   END;
   DateStamp(DSR);
   NminEND := SHORT (DSR.dsMinute);
   NticksEND := SHORT (DSR.dsTick);
   IF Mouvement ="R" THEN
      IF TempsMomentRec = 0
      THEN
        IF NminEND > NminBEG
        THEN NticksEND := NticksEND + (3000 * (NminEND - NminBEG))
        END;
        ent := NticksEND - NticksBEG;
        ree := real(ent);
        TempsMomentRec := entier (ree / 50.)

      END;
   ELSE
      IF TempsMomentDep = 0
      THEN
        IF NminEND > NminBEG
        THEN NticksEND := NticksEND + (3000 * (NminEND - NminBEG))
        END;
        ent := NticksEND - NticksBEG;
        ree := real(ent);
        TempsMomentDep := entier (ree / 50.)

      END;
   END;

END MomentDepense;


(*****************************************************************************
(*                              PosteDepense                                 *
(*****************************************************************************

PROCEDURE PosteDepense (VAR ValeurPoste : Str20);


   VAR phrase : ARRAY[0..40] OF CHAR;

   BEGIN

  DateStamp(DSR);
  NminBEG := SHORT (DSR.dsMinute);
  NticksBEG := SHORT (DSR.dsTick);
  Poste (ValeurPoste);
  DateStamp(DSR);
  NminEND := SHORT (DSR.dsMinute);
  NticksEND := SHORT (DSR.dsTick);

      IF TempsPosteDep = 0
      THEN
        IF NminEND > NminBEG
        THEN NticksEND := NticksEND + (3000 * (NminEND - NminBEG))
        END;
```

```
          ent := NticksEND - NticksBEG;
          ree := real(ent);
          TempsPosteDep := entier (ree / 50.)
        END;
        phrase := "Il donne le poste : ";
        ConcatString (phrase,ValeurPoste);
       IF TableauTraces.Indice <= 499 THEN
        CopyString (TableauTraces.Tab[TableauTraces.Indice],phrase);
        TableauTraces.Indice := TableauTraces.Indice + 1;
        END;
   END PosteDepense;

(*****************************************************************************
(*                              PosteRecette                                 *
(*****************************************************************************

PROCEDURE PosteRecette (VAR ValeurPoste : Str20);


   VAR phrase : ARRAY[0..40] OF CHAR;

   BEGIN
  DateStamp(DSR);
  NminBEG := SHORT (DSR.dsMinute);
  NticksBEG := SHORT (DSR.dsTick);
  PosteRec (ValeurPoste);
  DateStamp(DSR);
  NminEND := SHORT (DSR.dsMinute);
  NticksEND := SHORT (DSR.dsTick);
        IF TempsPosteRec = 0
        THEN
          IF NminEND > NminBEG
          THEN NticksEND := NticksEND + (3000 * (NminEND - NminBEG))
          END;
          ent := NticksEND - NticksBEG;
          ree := real(ent);
          TempsPosteRec := entier (ree / 50.);
        END;
        phrase := "Il donne le poste : ";
        ConcatString (phrase,ValeurPoste);
       IF TableauTraces.Indice <= 499 THEN
        CopyString (TableauTraces.Tab[TableauTraces.Indice],phrase);
        TableauTraces.Indice := TableauTraces.Indice + 1;
        END;
   END PosteRecette;


(*****************************************************************************
(*                              PorteMonnaie                                 *
(*****************************************************************************

PROCEDURE PorteMonnaie;
VAR I : INTEGER;
BEGIN
     I := 0;
     IF (TableauParametres [3] = 0) OR (TableauParametres [3] = 2)
     THEN SommeRestanteNombre (I);
        ELSE
            IF(TableauParametres [4] = 0) OR (TableauParametres [4] = 4)
                THEN SommeRestanteNombre (I)
                ELSE
                    IF((TableauParametres [4] = 1) OR
                       (TableauParametres [4] = 3) OR
                       (TableauParametres [4] = 5) OR
                       (TableauParametres [4] = 6))
                       THEN SommeRestanteBillets (I)
```

```
                    END
               END;
     END;

END PorteMonnaie;

(*****************************************************************************)
(*                                EPMRecette                                 *)
(*****************************************************************************)

PROCEDURE EPMRecette;
VAR I : INTEGER;
BEGIN
     I := 1;
     IF (TableauParametres [3] = 0) OR (TableauParametres [3] = 2)
     THEN SommeRestanteNombre (I);
        ELSE
            IF(TableauParametres [4] = 0) OR (TableauParametres [4] = 4)
                THEN SommeRestanteNombre (I)
                ELSE
                    IF((TableauParametres [4] = 1) OR
                       (TableauParametres [4] = 3) OR
                       (TableauParametres [4] = 5) OR
                       (TableauParametres [4] = 6))
                       THEN SommeRestanteBillets (I)
                    END
               END;
     END;

END EPMRecette;



(*****************************************************************************)
(*                                EPMDepense                                 *)
(*****************************************************************************)
PROCEDURE EPMDepense;
VAR I : INTEGER;
BEGIN
     I := 2;
     IF (TableauParametres [3] = 0) OR (TableauParametres [3] = 2)
        THEN SommeRestanteNombre (I);
        ELSE
            IF(TableauParametres [4] = 0) OR (TableauParametres [4] = 4)
                THEN SommeRestanteNombre (I)
                ELSE
                    IF((TableauParametres [4] = 1) OR
                       (TableauParametres [4] = 3) OR
                       (TableauParametres [4] = 5) OR
                       (TableauParametres [4] = 6))
                       THEN SommeRestanteBillets (I)
                    END
               END;
     END;

END EPMDepense;



(*****************************************************************************)
(*                      PresentationRecDep                                   *)
(*****************************************************************************)
PROCEDURE PresentationRecDep;

BEGIN
  PRD.PresentationRecDep
```

```
END PresentationRecDep;

(*****************************************************************************)
(*                        Correction Porte Monnaie                           *)
(*****************************************************************************)
PROCEDURE CorrectionPM;
VAR Phrase : ARRAY[0..40] OF CHAR;
    PhrasePron : ARRAY[0..99] OF CHAR;
    Trace : ARRAY[0..40] OF CHAR;
    SuiteTrace : ARRAY[0..10] OF CHAR;
BEGIN
  Phrase := "Combien avez-vous d'argent ?";
  PhrasePron := "Cobya ah vay voo  darjo ?";
  DateStamp(DSR);
  NminBEG := SHORT (DSR.dsMinute);
  NticksBEG := SHORT (DSR.dsTick);
  IF (TableauParametres[3] = 0)
  THEN CorrPMCL (Phrase,PhrasePron);
  END;
  IF (TableauParametres[3] = 1)
  THEN CorrPMBI (Phrase,PhrasePron);
  END;
  IF (TableauParametres[3] = 2)
  THEN CorrPMCA (Phrase,PhrasePron);
  END ;
  IF TableauParametres[11] = 1
  THEN  Reglette ("S",TRUE);
  END;
  DateStamp(DSR);
  NminEND := SHORT (DSR.dsMinute);
  NticksEND := SHORT (DSR.dsTick);
        IF TempsCorrection = 0
        THEN
          IF NminEND > NminBEG
          THEN NticksEND := NticksEND + (3000 * (NminEND - NminBEG))
          END;
          ent := NticksEND - NticksBEG;
          ree := real(ent);
          TempsCorrection := entier (ree / 50.);
        END;

  (* Trace de la somme initiale *)
   CopyString (Trace, "Il donne une s. initiale de : ");
   ConvRealToString (Sinit,SuiteTrace,2,Decimal);
   ConcatString (Trace,SuiteTrace);
   ConcatString (Trace," frs ");
       IF TableauTraces.Indice <= 499 THEN
   CopyString (TableauTraces.Tab[TableauTraces.Indice],Trace);
   TableauTraces.Indice := TableauTraces.Indice + 1;
       END;

END CorrectionPM;

END Interface.
```

```
IMPLEMENTATION MODULE ModPresentation;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM Rasters IMPORT SetRast,RastPortPtr;


CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};

VAR
  Done : BOOLEAN;
  Gl : GadgetPtr;
  Wp : WindowProc;
  Req : Requester;
  Sig : SignalSet;
  Msg : IntuiMessagePtr;
  Win,Won,Wun : WindowPtr;
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;


PROCEDURE GadgetHandler (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

BEGIN

  Done := TRUE;

END GadgetHandler;


PROCEDURE Presentation;

BEGIN

  WITH Wp DO
```

```
  procGadgetUp := GadgetHandler;
END;
Scr := CreateScreen (640,256,1,NIL);
IF (Scr # NIL) THEN
        cmap[00] := 0000H;
        cmap[01] := 0FFFH;
        cmap[02] := 0E00H;
        cmap[03] := 0A00H;
        cmap[04] := 0D80H;
        cmap[05] := 0FE0H;
        cmap[06] := 08F0H;
        cmap[07] := 0080H;
        cmap[08] := 00B6H;
        cmap[09] := 00DDH;
        cmap[10] := 00AFH;
        cmap[11] := 007CH;
        cmap[12] := 000FH;
        cmap[13] := 070FH;
        cmap[14] := 0C0EH;
        cmap[15] := 0C08H;
        cmap[16] := 0620H;
        cmap[17] := 0E52H;
        cmap[18] := 0A52H;
        cmap[19] := 0FCAH;
        cmap[20] := 0333H;
        cmap[21] := 0444H;
        cmap[22] := 0555H;
        cmap[23] := 0666H;
        cmap[24] := 0777H;
        cmap[25] := 0888H;
        cmap[26] := 0999H;
        cmap[27] := 0AAAH;
        cmap[28] := 0CCCH;
        cmap[29] := 0DDDH;
        cmap[30] := 0EEEH;
        cmap[31] := 0FFFH;
    LoadRGB4 (Scr^.ViewPort,ADR (cmap),2);
    BeginGadgetList();
    AddGadgetTextButton (1,1,ADR(" SUITE "));
    G1 := EndGadgetList();
    IF (G1 # NIL) THEN

    Win := CreateWindow (0,0,640,256,WIDCMP,WFlags2,NIL,Scr,NIL);
    Won := CreateWindow (0,0,640,256,WIDCMP,WFlags,NIL,Scr,NIL);
    Wun := CreateWindow (264,230,200,24,WIDCMP,WFlags2,G1,Scr,NIL);
    IF (Win # NIL) AND (Wun # NIL) AND (Won # NIL) THEN

     ActivateWindow (Won^);
     IF CreateConsole (Won^) THEN
           wSetCursor (Won^,FALSE);
           wMove (Won^,32,3);
           PutStr(Won^,ADR("GESTION DE COMPTES"));
           wMove (Won^,32,4);
           PutStr(Won^,ADR("------------------"));
           wMove (Won^,7,7);
           PutStr(Won^,ADR("Conception du logiciel :"));
           wMove (Won^,35,8);
           PutStr(Won^,ADR("Centre PSINHA"));
           wMove (Won^,35,9);
           PutStr(Won^,ADR("Département de psychologie, FUNDP (M.Mercier)")
           wMove (Won^,7,11);
           PutStr(Won^,ADR("Développement du logiciel :"));
           wMove (Won^,35,12);
           PutStr(Won^,ADR("Unité d'enseignement et de recherche"));
           wMove (Won^,35,13);
           PutStr(Won^,ADR("Institut d'informatique (M.Noirhomme)"));
```

```
            wMove (Won^,7,15);
            PutStr(Won^,ADR("Mémoire pour l'obtention du grade de"));
            wMove (Won^,7,16);
            PutStr(Won^,ADR("Licencié et Maître en Informatique"));
            wMove (Won^,35,18);
            PutStr(Won^,ADR("Olivier Berger"));
            wMove (Won^,35,19);
            PutStr(Won^,ADR("France Cheron"));
            wMove (Won^,35,20);
            PutStr(Won^,ADR("Marc Déplechin"));
            wMove (Won^,35,21);
            PutStr(Won^,ADR("Gianni Strappazzon"));
            wMove (Won^,7,23);
            PutStr(Won^,ADR("avec la collaboration du département de psycholog
e (A.Faton,M.Galand)"));
            wMove (Won^,7,24);
            PutStr(Won^,ADR("et de la fondation suisse des téléthèses (A.Baech
er,J.C.Gabu)"));
            DeleteConsole (Won^);
        END;

        Done := FALSE;
        WHILE (NOT Done) DO
         Sig := Wait(SignalSet{CARDINAL(Wun^.UserPort^.mpSigBit)});
         LOOP
           Msg := GetMsg(Wun^.UserPort^);
           IF (Msg = NIL) THEN EXIT; END;
           ProcIMsg (Wp, Msg);
         END;
        END;
        CloseWindow (Wun^);
        CloseWindow (Win^);
        CloseWindow (Won^);
      END;
      FreeGadgetList (Gl^);
      END;
      CloseScreen (Scr^);
    END;
END Presentation;

(*  BEGIN

    Presentation;    *)

END ModPresentation.
```

```
IMPLEMENTATION MODULE ModReglette;

FROM MathLib0 IMPORT entier;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Str20,Str40,TRecettes,TDepenses,
                              TableauParametres,Sinit,Max;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr,FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                              ConvStringToReal;
FROM Drawing IMPORT RectFill,SetAPen,Move,Draw;
FROM Text IMPORT Text;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};

VAR
  Done : BOOLEAN;
  Gl : GadgetPtr;
  Wp,WpOK : WindowProc;
  Sig : SignalSet;
  Msg : IntuiMessagePtr;
  Diff : REAL;
  Win,Win2 : WindowPtr;
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  long1,Couleur : INTEGER;
  MaxStr,StrMont,StrPhr : Str40;
  bou,Indice,long,FinRec,DebRec,long2,DebDep,FinDep,Abs,indice,i : INTEGER;
  TotDep,TotRec,TotDiff : REAL;
  Rp : RastPortPtr;
  Minicone : ARRAY [0..2] OF Image;
  ImgPtr : ImageDescTablePtr;
  ImgCount : CARDINAL;
```

```
PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

BEGIN

  Done := TRUE;

END GadgetHandlerOK;

PROCEDURE fleched (ord,co : INTEGER);

BEGIN

  SetAPen (Rp^,co);
  RectFill (Rp^,227,ord + 5,232,ord +7);
  Move (Rp^,230,ord + 3);
  Draw (Rp^,233,ord + 6);
  Draw (Rp^,230,ord + 9);
  SetAPen (Rp^,1);

END fleched;

PROCEDURE flecheg (ord,co : INTEGER);

BEGIN

  SetAPen (Rp^,co);
  RectFill (Rp^,17,ord + 5,23,ord +7);
  Move (Rp^,19,ord + 3);
  Draw (Rp^,16,ord + 6);
  Draw (Rp^,19,ord + 9);
  SetAPen (Rp^,1);

END flecheg;

PROCEDURE Clignoter (debut,fin,ordonnee : INTEGER);

BEGIN

  IF (debut<=225)
  THEN
    IF (debut >25)
    THEN
      IF (fin > 225)
      THEN
        i := 0;
        WHILE i <= 3 DO
         SetAPen (Rp^,0);
         RectFill (Rp^,debut,ordonnee,225,ordonnee + 12);
         fleched (ordonnee,0);
         Delay (50);
         SetAPen (Rp^,Couleur);
         RectFill (Rp^,debut,ordonnee,225,ordonnee + 12);
         fleched (ordonnee,Couleur);
         Delay (50);
         i := i + 1;
        END;
      ELSE
        i := 0;
        WHILE i <= 3 DO
          SetAPen (Rp^,0);
          RectFill (Rp^,debut,ordonnee,fin,ordonnee + 12);
          Delay (50);
          SetAPen (Rp^,Couleur);
          RectFill (Rp^,debut,ordonnee,fin,ordonnee + 12);
          Delay (50);
          i := i + 1;
```

```
        END;
      END;
    ELSE
      IF fin > 225
      THEN
        i := 0;
        WHILE i <= 3 DO
         SetAPen (Rp^,0);
         RectFill (Rp^,26,ordonnee,225,ordonnee + 12);
         fleched (ordonnee,0);
         flecheg (ordonnee,0);
         Delay (50);
         SetAPen (Rp^,Couleur);
         RectFill (Rp^,26,ordonnee,225,ordonnee + 12);
         fleched (ordonnee,Couleur);
         flecheg (ordonnee,Couleur);
         Delay (50);
         i := i + 1;
        END;
      ELSE
        i := 0;
        WHILE i <= 3 DO
          SetAPen (Rp^,0);
          RectFill (Rp^,26,ordonnee,fin,ordonnee + 12);
          flecheg (ordonnee,0);
          Delay (50);
          SetAPen (Rp^,Couleur);
          RectFill (Rp^,26,ordonnee,fin,ordonnee + 12);
          flecheg (ordonnee,Couleur);
          Delay (50);
          i := i + 1;
        END;
      END;
    END;
  ELSE
    i := 0;
    WHILE i <=3 DO
      SetAPen (Rp^,0);
      fleched (ordonnee,0);
      Delay (50);
      fleched (ordonnee,Couleur);
      Delay (50);
      i := i + 1;
    END;
  END;
  SetAPen (Rp^,1);

END Clignoter;

PROCEDURE CalcRec (ind : INTEGER; VAR TotRec : REAL; VAR deb : INTEGER);

VAR
  i ,long1 : INTEGER;

BEGIN
  i := 0;
  TotRec := 0.;
  WHILE (i<=ind) DO
   TotRec := TotRec + TRecettes.Tab[i].Valeur;
   i := i + 1;
  END;
  long1 := entier ((TotRec/Max)*200.);
  IF (long1 = 0) AND (TotRec > 0.)
  THEN long1 := 1;
  END;
  deb := 26 + long1;
```

```
  IF long1 > 0
  THEN
    IF long1 > 200
    THEN
       long1 := 200;
       fleched (97,8);
    END;
    SetAPen (Rp^,8);
    RectFill (Rp^,26,97,25 + long1,109);
    SetAPen (Rp^,1);
  END;

END CalcRec;

PROCEDURE CalcDep (ind : INTEGER; VAR TotDep : REAL; VAR deb : INTEGER);

VAR
  i ,long1,g,d : INTEGER;

BEGIN
  i := 0;
  TotDep := 0.;
  WHILE (i<=ind) DO
   TotDep := TotDep + TDepenses.Tab[i].Valeur;
   i := i + 1;
  END;
  long1 := entier ((TotDep/Max)*200.);
  IF (long1 = 0) AND (TotDep > 0.)
  THEN long1 := 1;
  END;
  deb := FinRec - long1;
  g := deb + 1;
  d := FinRec;
  IF long1 > 0
  THEN
    IF g <= 225
    THEN
     IF g < 26
     THEN
       flecheg (129,2);
       g := 26;
     END;
     IF FinRec > 225
     THEN
       d := 225;
       fleched (129,2);
     END;
     SetAPen (Rp^,2);
     RectFill (Rp^,g,129,d,141);
     SetAPen (Rp^,1);
    END;
  END;

END CalcDep;

PROCEDURE CalcDiff (TotRec,TotDep : REAL; co : INTEGER);

VAR
  long1 : INTEGER;
  StrDiff : Str20;
  TotDiff : REAL;

BEGIN
  TotDiff := TotRec - TotDep;
  long1 := entier ((TotDiff/Max)*200.);
  IF (long1 = 0) AND (TotDiff <> 0.)
```

```
  THEN
   IF TotDiff < 0.
   THEN
     long1 := -1;
   ELSE
     long1 := 1;
   END;
  END;
  SetAPen (Rp^,co);
  IF long1 < 0
  THEN
    flecheg (161,co);
  ELSE
    IF long1 > 200
    THEN
       long1 := 200;
       fleched (161,co);
    END;
    SetAPen (Rp^,co);
    RectFill (Rp^,26,161,25 + long1,173);
  END;
  ConvRealToString (TotDiff,StrDiff,0,Decimal);
  SetAPen (Rp^,co);
  Move (Rp^,240,171);
  Text (Rp^,ADR(StrDiff),StringLength(StrDiff));
  SetAPen (Rp^,1);

END CalcDiff;


PROCEDURE CalcSI (co : INTEGER);

VAR
  StrInit : Str20;
  long1 : INTEGER;

BEGIN
  long1 := entier ((Sinit/Max)*200.);
  IF (long1 = 0) AND (Sinit > 0.)
  THEN long1 := 1;
  END;
  SetAPen (Rp^,co);
  IF long1 > 0
  THEN
    IF long1 > 200
    THEN
       long1 := 200;
       fleched (193,co);
    END;
    SetAPen (Rp^,co);
    RectFill (Rp^,26,193,25 + long1,205);
  END;
  ConvRealToString (Sinit,StrInit,0,Decimal);
  Move (Rp^,240,203);
  Text (Rp^,ADR(StrInit),StringLength(StrInit));
  SetAPen (Rp^,1);

END CalcSI;


PROCEDURE Reglette (Orig_Mvt : CHAR; Affectation:BOOLEAN);

BEGIN

(*          Max := 2000.;
  Sinit := 500.;
```

```
TRecettes.Tab[0].Valeur := 600.;
TRecettes.Tab[1].Valeur := 200.;
TRecettes.Tab[2].Valeur := 100.;
TRecettes.Tab[3].Valeur := 0.;
TDepenses.Tab[0].Valeur := 1.;
TDepenses.Tab[1].Valeur := 100.;

TRecettes.Indice := 1;
TDepenses.Indice := 2;              *)



WITH WpOK DO
  procGadgetUp := GadgetHandlerOK;
END;

Scr := CreateScreen (320,256,5,NIL);
IF (Scr # NIL) THEN
        cmap[00] := 0000H;
        cmap[01] := 0FFFH;
        cmap[02] := 0E00H;
        cmap[03] := 0A00H;
        cmap[04] := 0D80H;
        cmap[05] := 0FE0H;
        cmap[06] := 0FCAH;
        cmap[07] := 0080H;
        cmap[08] := 00B6H;
        cmap[09] := 00DDH;
        cmap[10] := 00AFH;
        cmap[11] := 007CH;
        cmap[12] := 000FH;
        cmap[13] := 070FH;
        cmap[14] := 0C0EH;
        cmap[15] := 0C08H;
        cmap[16] := 0620H;
        cmap[17] := 0E52H;
        cmap[18] := 0A52H;
        cmap[19] := 0FCAH;
        cmap[20] := 0333H;
        cmap[21] := 0444H;
        cmap[22] := 0555H;
        cmap[23] := 0666H;
        cmap[24] := 0777H;
        cmap[25] := 0888H;
        cmap[26] := 0999H;
        cmap[27] := 0AAAH;
        cmap[28] := 0CCCH;
        cmap[29] := 0DDDH;
        cmap[30] := 0EEEH;
        cmap[31] := 0FFFH;
    LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

    ImgPtr := FindImageTable(16111111H, ImgCount);
    IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
      WHILE indice <= 2 DO
        WITH Minicone[indice] DO
          LeftEdge    := 0;
          TopEdge     := 0;
          Width       := ImgPtr^[indice].Width;
          Height      := ImgPtr^[indice].Height;
          Depth       := ImgPtr^[indice].Depth;
          ImageData   := ImgPtr^[indice].Data;
          PlanePick   := BYTE(31);
          PlaneOnOff  := BYTE(31);
          NextImage := NIL;
        END;
```

```
    indice := indice + 1;
  END;
END;


BeginGadgetList();
AddGadgetTextButton (1,1,ADR("SUITE"));
G1 := EndGadgetList();
IF (G1 # NIL) THEN

Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win # NIL) THEN
   Rp := Win^.RPort;
   SetAPen (Rp^,1);
   Move (Rp^,0,0);
   Draw (Rp^,319,0);
   Draw (Rp^,319,255);
   Draw (Rp^,0,255);
   Draw (Rp^,0,0);
   Move (Rp^,0,53);
   Draw (Rp^,319,53);
   Couleur := 0;
   IF Orig_Mvt = "R"
   THEN Couleur := 8;
   END;
   IF Orig_Mvt = "D"
   THEN Couleur := 2;
   END;
   IF Orig_Mvt = "S"
   THEN Couleur := 5;
   END;
   SetAPen(Rp^,Couleur);
   Move (Rp^,1,1);
   Draw (Rp^,318,1);
   Draw (Rp^,318,52);
   Draw (Rp^,1,52);
   Draw (Rp^,1,1);
   Move (Rp^,2,2);
   Draw (Rp^,317,2);
   Draw (Rp^,317,51);
   Draw (Rp^,2,51);
   Draw (Rp^,2,2);
   SetAPen (Rp^,1);
   Move (Rp^,26,70);
   Draw (Rp^,225,70);
   Draw (Rp^,225,68);
   Draw (Rp^,225,72);
   Move (Rp^,26,68);
   Draw (Rp^,26,72);
   Move (Rp^,110,65);
   ConvRealToString (Max,MaxStr,0,Decimal);
   Text (Rp^,ADR(MaxStr),StringLength(MaxStr));
   Move (Rp^,25,77);
   Draw (Rp^,25,230);
   DrawImage (Rp^,Minicone [0],290,92);
   DrawImage (Rp^,Minicone [1],290,124);
   DrawImage (Rp^,Minicone [2],290,188);
   Move (Rp^,300,168);
   SetAPen (Rp^,8);
   Text (Rp^,ADR("R"),StringLength("R"));
   Move (Rp^,292,178);
   SetAPen (Rp^,1);
   Text (Rp^,ADR("-"),StringLength("-"));
   Move (Rp^,300,178);
   SetAPen (Rp^,2);
   Text (Rp^,ADR("D"),StringLength("D"));
```

```
IF (Orig_Mvt = "R") AND (Affectation = TRUE)
THEN
  IF (TRecettes.Indice = 1)
  THEN
    SetAPen (Rp^,0);
    RectFill(Rp^,266,124,316,220);
    SetAPen (Rp^,1);
    Move (Rp^,7,12);
    ConvRealToString (TRecettes.Tab[0].Valeur,StrMont,0,Decimal);
    CopyString (StrPhr,"La dernière fois, il vous restait ");
    Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
    Move (Rp^,7,26);
    CopyString (StrPhr,StrMont);
    ConcatString (StrPhr," Francs ");
    ConcatString (StrPhr,"dans votre porte-monnaie");
    Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
    Move (Rp^,240,107);
    Text (Rp^,ADR(StrMont),StringLength(StrMont));
    long := entier ((TRecettes.Tab[0].Valeur / Max) * 200.);
    IF (long=0) AND (TRecettes.Tab[0].Valeur > 0.)
    THEN
       long := 1;
    END;
    SetAPen (Rp^,8);
    IF long >0
    THEN
     SetAPen (Rp^,8);
     IF long > 200
     THEN RectFill (Rp^,26,35,225,47);
          fleched (35,8);
     ELSE
          RectFill (Rp^,26,35,25 + long,47);
     END;
     Clignoter (26,25+long,97);
    END;
  ELSE
    SetAPen (Rp^,1);
    Move (Rp^,20,20);
    ConvRealToString (TRecettes.Tab[TRecettes.Indice - 1].Valeur,
                      StrMont,0,Decimal);
    CopyString (StrPhr,"Cette recette s'élève à ");
    ConcatString (StrPhr,StrMont);
    ConcatString (StrPhr," Francs.");
    Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
    CalcRec(TRecettes.Indice - 2,TotRec,DebRec);
    TotRec := TotRec + TRecettes.Tab[TRecettes.Indice - 1].Valeur;
    ConvRealToString (TotRec,StrMont,0,Decimal);
    Move (Rp^,240,107);
    SetAPen (Rp^,8);
    Text (Rp^,ADR(StrMont),StringLength(StrMont));
    SetAPen (Rp^,1);
    long := entier ((TRecettes.Tab[TRecettes.Indice - 1].Valeur /Max)
                    * 200.);
    IF (long = 0) AND (TRecettes.Tab[TRecettes.Indice - 1].Valeur >0.
    THEN long :=1;
    END;
    long2 := long;
    IF (long > 0)
    THEN
     SetAPen (Rp^,8);
     IF long > 200
     THEN long := 200;
          fleched (35,8);
     END;
     SetAPen (Rp^,8);
     RectFill (Rp^,26,35,25+long,47);
```

```
      END;
      FinRec := DebRec + long2 -1;
      CalcDep (TDepenses.Indice-1,TotDep,DebDep);
      ConvRealToString (TotDep, StrMont,0,Decimal);
      Move (Rp^,240,139);
      SetAPen (Rp^,2);
      Text (Rp^,ADR(StrMont),StringLength(StrMont));
      SetAPen (Rp^,1);
      CalcDiff (TotRec,TotDep,6);
      CalcSI (5);
      Clignoter (DebRec,FinRec,97);
    END;
  END;

  IF (Orig_Mvt = "D") AND (Affectation = TRUE)
  THEN
      SetAPen (Rp^,1);
      Move (Rp^,20,20);
      ConvRealToString (TDepenses.Tab[TDepenses.Indice-1].Valeur,
                        StrMont,0,Decimal);
      CopyString (StrPhr,"Cette dépense s'élève à ");
      ConcatString (StrPhr,StrMont);
      ConcatString (StrPhr," Francs.");
      Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
      CalcRec(TRecettes.Indice - 1,TotRec,DebRec);
      ConvRealToString (TotRec,StrMont,0,Decimal);
      Move (Rp^,240,107);
      SetAPen (Rp^,8);
      Text (Rp^,ADR(StrMont),StringLength(StrMont));
      SetAPen (Rp^,1);
      FinRec := DebRec - 1;
      CalcDep(TDepenses.Indice - 2,TotDep,DebDep);
      TotDep := TotDep + TDepenses.Tab [TDepenses.Indice-1].Valeur;
      ConvRealToString (TotDep,StrMont,0,Decimal);
      Move (Rp^,240,139);
      SetAPen (Rp^,2);
      Text (Rp^,ADR(StrMont),StringLength(StrMont));
      long := entier ((TDepenses.Tab[TDepenses.Indice-1].Valeur /Max)
                      * 200.);
      IF (long = 0) AND (TDepenses.Tab[TDepenses.Indice-1].Valeur >0.)
      THEN long :=1;
      END;
      long2 := long;
      IF (long > 0)
      THEN
       SetAPen (Rp^,2);
       IF long > 200
       THEN long := 200;
            fleched (35,2);
       END;
       SetAPen (Rp^,2);
       RectFill (Rp^,26,35,25+long,47);
      END;
      SetAPen (Rp^,1);
      CalcDiff (TotRec,TotDep,6);
      CalcSI (5);
      IF DebDep > 25
      THEN
         Clignoter (DebDep - long2,DebDep,129);
      END;
  END;
  IF (Orig_Mvt ="S") AND (Affectation = TRUE)
  THEN
      SetAPen (Rp^,1);
      Move (Rp^,30,12);
      ConvRealToString (Sinit,StrMont,0,Decimal);
```

```
        CopyString (StrPhr,"Vous avez ");
        ConcatString (StrPhr,StrMont);
        ConcatString (StrPhr," Francs");
        Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
        Move (Rp^,30,26);
        Text (Rp^,ADR("dans votre porte-monnaie"),
                  StringLength("dans votre porte-monnaie"));
        SetAPen (Rp^,5);
        Move (Rp^,240,203);
        Text (Rp^,ADR(StrMont),StringLength(StrMont));
        SetAPen (Rp^,1);
        long := entier ((Sinit /Max)* 200.);
        IF (long = 0) AND (Sinit >0.)
        THEN long :=1;
        END;
        long2 := long;
        IF (long > 0)
        THEN
         SetAPen (Rp^,5);
         IF long > 200
         THEN long := 200;
              fleched (35,5);
         END;
         SetAPen (Rp^,5);
         RectFill (Rp^,26,35,25+long,47);
        END;
        CalcRec(TRecettes.Indice - 1,TotRec,DebRec);
        ConvRealToString (TotRec,StrMont,0,Decimal);
        Move (Rp^,240,107);
        SetAPen (Rp^,8);
        Text (Rp^,ADR(StrMont),StringLength(StrMont));
        FinRec := DebRec -1 ;
        CalcDep (TDepenses.Indice-1,TotDep,DebDep);
        ConvRealToString (TotDep, StrMont,0,Decimal);
        Move (Rp^,240,139);
        SetAPen (Rp^,2);
        Text (Rp^,ADR(StrMont),StringLength(StrMont));
        SetAPen (Rp^,1);
        CalcDiff (TotRec,TotDep,6);
        IF long2 > 0
        THEN
          Clignoter (26,25 + long2,193);
        END;
    END;
    IF (Orig_Mvt = "R") AND (Affectation = FALSE)
    THEN
        i := 0;
        TotRec := 0.;
        WHILE (i<=TRecettes.Indice - 1) DO
             TotRec := TotRec + TRecettes.Tab[i].Valeur;
             i := i + 1;
        END;
        long1 := entier ((TotRec/Max)*200.);
        IF (long1 = 0) AND (TotRec > 0.)
        THEN long1 := 1;
        END;
        DebRec := 26 + long1;
        SetAPen (Rp^,1);
        Move (Rp^,30,12);
        ConvRealToString (TotRec,StrMont,0,Decimal);
        Text (Rp^,ADR("Le total de vos recettes"),
                  StringLength("Le total de vos recettes"));
        Move (Rp^,30,26);
        CopyString (StrPhr,"est de ");
        ConcatString (StrPhr,StrMont);
        ConcatString (StrPhr," Francs.");
```

```
Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
Move (Rp^,240,107);
SetAPen (Rp^,8);
Text (Rp^,ADR(StrMont),StringLength(StrMont));
SetAPen (Rp^,1);
long := entier ((TotRec /Max)* 200.);
IF (long = 0) AND (TotRec >0.)
THEN long :=1;
END;
long2 := long;
IF (long > 0)
THEN
 SetAPen (Rp^,8);
 IF long > 200
 THEN long := 200;
      flecheD (35,8);
 END;
 SetAPen (Rp^,8);
 RectFill (Rp^,26,35,25+long,47);
END;
FinRec := DebRec -1;
CalcDep (TDepenses.Indice-1,TotDep,DebDep);
ConvRealToString (TotDep, StrMont,0,Decimal);
Move (Rp^,240,139);
SetAPen (Rp^,2);
Text (Rp^,ADR(StrMont),StringLength(StrMont));
SetAPen (Rp^,1);
CalcDiff (TotRec,TotDep,6);
CalcSI (5);
Abs := 26;
Indice := 0;
WHILE Indice < TRecettes.Indice DO
  long1 := entier ((TRecettes.Tab[Indice].Valeur /Max)*200.);
  IF (long1 = 0) AND (TRecettes.Tab[Indice].Valeur > 0.)
  THEN long1 := 1;
  END;
  bou := Abs + long1 - 1;
  IF long1 > 0
  THEN
    IF bou >225
    THEN
      IF Abs <= 225
      THEN
         bou := 225;
         flecheD (97,8);
         SetAPen (Rp^,8);
         RectFill (Rp^,Abs,97,bou,109);
         Delay (60);
         flecheD (97,0);
         SetAPen (Rp^,0);
         RectFill (Rp^,Abs,97,bou,109);
         Delay (60);
         flecheD (97,8);
         SetAPen (Rp^,8);
         RectFill (Rp^,Abs,97,bou,109);
         Delay (60);
      ELSE
         flecheD (97,8);
         Delay (60);
         flecheD (97,0);
         Delay (60);
         flecheD (97,8);
         Delay (60);
      END;
    ELSE
         SetAPen (Rp^,8);
```

```
                RectFill (Rp^,Abs,97,bou,109);
                Delay (60);
                SetAPen (Rp^,0);
                RectFill (Rp^,Abs,97,bou,109);
                Delay (60);
                SetAPen (Rp^,8);
                RectFill (Rp^,Abs,97,bou,109);
                Delay (60);
          END;
        END;
        SetAPen (Rp^,1);
        Abs := Abs + long1;
        Indice := Indice + 1;
      END;
      SetAPen (Rp^,1);
  END;
  IF (Orig_Mvt = "D") AND (Affectation = FALSE)
  THEN
      CalcRec (TRecettes.Indice - 1,TotRec,DebRec);
      SetAPen (Rp^,8);
      Move (Rp^,240,107);
      ConvRealToString (TotRec,StrMont,0,Decimal);
      Text (Rp^,ADR(StrMont),StringLength(StrMont));
      FinRec := DebRec - 1;
      i := 0;
      TotDep := 0.;
      WHILE (i<=TDepenses.Indice-1) DO
         TotDep := TotDep + TDepenses.Tab[i].Valeur;
         i := i + 1;
      END;
      long1 := entier ((TotDep/Max)*200.);
      IF (long1 = 0) AND (TotDep > 0.)
      THEN long1 := 1;
      END;
      DebRec := FinRec - long1;
      ConvRealToString (TotDep,StrMont,0,Decimal);
      Move (Rp^,240,139);
      SetAPen (Rp^,2);
      Text (Rp^,ADR(StrMont),StringLength(StrMont));
      SetAPen(Rp^,1);
      Move (Rp^,30,12);
      Text (Rp^,ADR("Le total de vos dépenses"),
                StringLength("Le total de vos dépenses"));
      Move (Rp^,30,26);
      CopyString (StrPhr,"est de ");
      ConcatString (StrPhr,StrMont);
      ConcatString (StrPhr," Francs.");
      Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
      long := entier ((TotDep /Max)* 200.);
      IF (long = 0) AND (TotDep >0.)
      THEN long :=1;
      END;
      long2 := long;
      IF (long > 0)
      THEN
       SetAPen (Rp^,2);
       IF long > 200
       THEN long := 200;
            fleched (35,2);
       END;
       SetAPen (Rp^,2);
       RectFill (Rp^,26,35,25+long,47);
      END;
      SetAPen (Rp^,1);
      CalcDiff (TotRec,TotDep,6);
      CalcSI (5);
```

```
Indice := 0;
WHILE Indice < TDepenses.Indice DO
  long1 := entier ((TDepenses.Tab[Indice].Valeur /Max)*200.);
  IF (long1 = 0) AND (TDepenses.Tab[Indice].Valeur > 0.)
  THEN long1 := 1;
  END;
  DebRec := FinRec - long1 + 1;
    IF (FinRec > 225) AND (DebRec > 225)
    THEN
         fleched (129,2);
         Delay (60);
         fleched (129,0);
         Delay (60);
         fleched (129,2);
         Delay (60);
    END;
    IF (FinRec > 225) AND (DebRec <= 225) AND
                          ((FinRec - long1 + 1) >= 26)
    THEN
         fleched (129,2);
         SetAPen (Rp^,2);
         RectFill (Rp^,DebRec,129,225,141);
         Delay (60);
         fleched (129,0);
         SetAPen (Rp^,0);
         RectFill (Rp^,DebRec,129,225,141);
         Delay (60);
         fleched (129,2);
         SetAPen (Rp^,2);
         RectFill (Rp^,DebRec,129,225,141);
         Delay (60);
    END;
    IF (FinRec > 225) AND (DebRec < 26)
    THEN
         fleched (129,2);
         flecheg (129,2);
         SetAPen (Rp^,2);
         RectFill (Rp^,26,129,225,141);
         Delay (60);
         fleched (129,0);
         flecheg (129,0);
         SetAPen (Rp^,0);
         RectFill (Rp^,26,129,225,141);
         Delay (60);
         fleched (129,2);
         flecheg (129,2);
         SetAPen (Rp^,2);
         RectFill (Rp^,26,129,225,141);
         Delay (60);
    END;
    IF (FinRec <=225) AND (DebRec < 26)
    THEN
         flecheg (129,2);
         SetAPen (Rp^,2);
         RectFill (Rp^,26,129,FinRec,141);
         Delay (60);
         flecheg (129,0);
         SetAPen (Rp^,0);
         RectFill (Rp^,26,129,FinRec,141);
         Delay (60);
         flecheg (129,2);
         SetAPen (Rp^,2);
         RectFill (Rp^,26,129,FinRec,141);
         Delay (60);
    END;
    IF (FinRec <=225) AND (DebRec >=25) AND (DebRec <=225)
```

```
          THEN
               SetAPen (Rp^,2);
               RectFill (Rp^,DebRec,129,FinRec,141);
               Delay (60);
               SetAPen (Rp^,0);
               RectFill (Rp^,DebRec,129,FinRec,141);
               Delay (60);
               SetAPen (Rp^,2);
               RectFill (Rp^,DebRec,129,FinRec,141);
               Delay (60);
          END;
          FinRec := DebRec - 1 ;
          Indice := Indice + 1;

     END;
     SetAPen (Rp^,1);
END;
IF (Orig_Mvt = "T")
THEN
   CalcRec (TRecettes.Indice - 1,TotRec,DebRec);
   ConvRealToString (TotRec,StrMont,0,Decimal);
   Move (Rp^,240,107);SetAPen (Rp^,8);
   Text (Rp^,ADR(StrMont),StringLength(StrMont));
   FinRec := DebRec - 1;
   CalcDep (TDepenses.Indice-1,TotDep,DebDep);
   ConvRealToString (TotDep,StrMont,0,Decimal);
   Move (Rp^,240,139);SetAPen (Rp^,2);
   Text (Rp^,ADR(StrMont),StringLength(StrMont));
   TotDiff := TotRec - TotDep;
   IF TotDiff > Sinit
   THEN Diff := TotDiff - Sinit;
   ELSE Diff := Sinit - TotDiff;
   END;
   IF (Diff < 0.5)
   THEN Diff := 0.;
   END;
   SetAPen (Rp^,1);
   IF (Diff = 0.)
   THEN
     CopyString (StrPhr,"Bravo, il y a équilibre des comptes");
     Move (Rp^,25,20);
     Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
   ELSE
     ConvRealToString (Diff,StrMont,0,Decimal);
     CopyString (StrPhr,"Il y a un déséquilibre");
     Move (Rp^,25,12);
     Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
     CopyString (StrPhr,"de ");
     ConcatString (StrPhr,StrMont);
     ConcatString (StrPhr," Francs.");
     Move (Rp^,25,26);
     Text (Rp^,ADR(StrPhr),StringLength(StrPhr));
     long := entier ((Diff /Max) *200.);
     IF (long = 0) AND (Diff > 0.)
     THEN long := 1;
     END;
     IF (long > 0)
     THEN
       SetAPen (Rp^,5);
       IF long > 200
       THEN long := 200;
            fleched (35,5);
       END;
       SetAPen (Rp^,5);
       RectFill (Rp^,26,35,25 + long,47);
     END;
```

```
            END;
            i := 1;
            WHILE i <= 3 DO
              CalcDiff (TotRec,TotDep,0);
              CalcSI (0);
              Delay (50);
              CalcDiff (TotRec,TotDep,6);
              CalcSI (5);
              Delay (50);
              i := i+ 1;
            END;
          END;

       Win2 := CreateWindow (132,237,180,15,WIDCMP,WFlags2,Gl,Scr,NIL);
       IF Win2 # NIL THEN
       Done := FALSE;
       WHILE (NOT Done) DO
        Sig := Wait(SignalSet{CARDINAL(Win2^.UserPort^.mpSigBit)});
        LOOP
          Msg := GetMsg(Win2^.UserPort^);
          IF (Msg = NIL) THEN EXIT; END;
          ProcIMsg (WpOK, Msg);
        END;
       END;
       CloseWindow (Win2^);
       END;
      CloseWindow(Win^);
      END;
      FreeGadgetList (Gl^);
      END;
      CloseScreen(Scr^);
      END;

END Reglette;

(*   VAR
     o : CHAR;
     a : BOOLEAN;

BEGIN
     o := "T";
     a := TRUE;
     Reglette (o,a);          *)

END ModReglette.
```

```
IMPLEMENTATION MODULE CorrectionPMCA;

FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton,AddGadgetImageButton,GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine;
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
       StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3,Str9,Str20,Str40, TabNombre,
   TableauTraces,TRecettes,TDepenses,TabNombreInit,Max,Sactuel,Sinit,Str100,
   TableauParametres,TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                  ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Utilitaires IMPORT ConvPrononcable;
FROM RealInOut IMPORT WriteReal;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet éGadgetUpè;
  WFlags = WindowFlagsSet éActivateè;
  WFlags2 = WindowFlagsSet éActivate,Borderlessè;
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre,DoneRec,Recom : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff,QR : BOOLEAN;
  ActivPoint, DoneCor, Corri : BOOLEAN;
  For : RealToStringFormat;
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Win3,Win4,Wun : WindowPtr;
```

```
  TabImages: ARRAY °0..59§ OF Image;
  ImgPtr,ImgPtr2,ImgPtr3,ImgPtr4,ImgPtr5 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,ImgCount4,ImgCount5,
                                        Indice,list6,list7, i : CARDINAL;
  cmap : ARRAY °0..31§ OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK,GlRec,GlCor : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK,WpRec,WpCor : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK,ReqRec,ReqCor : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK,MsgRec,MsgCor : IntuiMessagePtr;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005,
  IRecettes,IDepenses,Fait,I : INTEGER;
  Somme, SommePrecedente, SommeMax ,SA,SAP: REAL;
  ES,fonc,Maxaf : Str20;
  Res,Quit : BOOLEAN;
  DerPron,PPron : Str100;
  StrSomme : Str20;
  CompChiffre : INTEGER;
  OKS : BOOLEAN;
  Snouve,Snouvede,Zero : REAL;
  DSR : DateStampRecord;
  ActMinute,ActTick,ActTick2,ent : INTEGER;
  ree : REAL;
  Rp : RastPortPtr;

PROCEDURE Reel (VAR St:ARRAY OF CHAR; VAR OK :BOOLEAN);

VAR Indice : CARDINAL;
    Premier: BOOLEAN;

BEGIN
    Indice :=0;
    OK :=TRUE;
    Premier :=TRUE;
    IF (CompareString(St,"") = equal ) THEN OK:=FALSE END;
    IF (CompareString(St,"-")= equal ) THEN OK:=FALSE END;
    WHILE ((Indice<StringLength(St)) AND OK ) DO
     IF (( St°Indice§<"0") OR (St°Indice§>"9")) THEN
         IF (((St°Indice§=",")OR(St°Indice§=".")) AND (Premier)) THEN
             St°Indice§:=".";
             Indice :=Indice+1;
             Premier:=FALSE;
         ELSE
           IF ((St°Indice§="-") AND (Indice=StringLength(St)-1)) THEN
             St°Indice§:=" ";
             Indice := Indice+1;
           ELSE OK := FALSE
           END;
         END;
     ELSE
         Indice := Indice+1
     END;
    END;
END Reel;


PROCEDURE GadgetHandlerFinCA (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Fini := TRUE ; DoneFin := TRUE;
           IF TableauChaine.Indice < 499
           THEN TableauChaine.Tab°TableauChaine.Indice§ :=29;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
```

```
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
            TableauChaine.Tab°TableauChaine.Indice§ :=7;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
        END;
        Trace := "Il confirme la demande ";
        IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END; ù

1 : Fini := FALSE; DoneFin := TRUE;
        IF TableauChaine.Indice < 499
        THEN TableauChaine.Tab°TableauChaine.Indice§ :=29;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
            TableauChaine.Tab°TableauChaine.Indice§ :=26;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
        END;
        Trace := "Il infirme la demande ";
        Somme := 0.;
        IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
```

```
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; ù
      2 :
  END;
END GadgetHandlerFinCA;


PROCEDURE GadgetHandlerRecCA (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Recom := TRUE ; DoneRec := TRUE;
           Trace := "Il confirme la demande ";
           CopyString (StrSomme,"");
           ActivPoint := FALSE;
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END; ù
     1 : Recom := FALSE; DoneRec := TRUE;
           Trace := "Il infirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END; ù
     2 :
  END;
END GadgetHandlerRecCA;

PROCEDURE GadgetHandlerCorCA (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Corri := TRUE ; DoneCor := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab°TableauChaine.Indice§ :=26;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
           TableauChaine.Indice := TableauChaine.Indice + 1;
           END;
           Trace := "Il demande à corriger";
           CopyString (StrSomme,"");
           ActivPoint := FALSE;
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END; ¦
     1 : Corri := FALSE; DoneCor := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab[TableauChaine.Indice] :=7;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
```

```
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il ne corrige rien ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
    2 :
  END;
END GadgetHandlerCorCA;

PROCEDURE GadgetHandlerCA (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
    SomAct : REAL;
BEGIN
  IF OpenSpeech () THEN
  CASE Gad.GadgetID OF
    0 : ActivEff := TRUE;

        IF (CompChiffre <=8) THEN
         IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
        CompChiffre := CompChiffre + 1;
        IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("zero"));
           DerPron := "zero";
        END;
        wMove (Won^,1,1);
        wClrEndLine (Won^);
        ConcatString (StrSomme,"0");
        wMove (Won^,2,1);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,12,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        Trace := "Il appuie sur 0 ";
        IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
        END;
        END |

    1 : ActivEff := TRUE;

        IF (CompChiffre <=8) THEN
         IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
        CompChiffre := CompChiffre + 1;
        IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("an"));
           DerPron := "an";
        END;
        wMove (Won^,1,1);
        wClrEndLine (Won^);
        ConcatString (StrSomme,"1");
        wMove (Won^,2,1);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,12,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        Trace := "Il appuie sur 1 ";
        IF TableauTraces.Indice <= 499 THEN
```

```
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END ;

2 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("duh"));
       DerPron := "duh";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"2");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 2 Frs ";
    IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END ;

3 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("trwa"));
       DerPron := "trwa";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"3");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
            Trace := "Il appuie sur 3 ";
    IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END ;

4 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("katr"));
       DerPron := "katr";
    END;
    wMove (Won^,1,1);
```

```
    wClrEndLine (Won^);
    ConcatString (StrSomme,"4");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 4 ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END |

5 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("senk"));
       DerPron := "senk";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"5");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 5 ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END |

6 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("sys"));
       DerPron := "sys";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"6");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 6 ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END |

7 : ActivEff := TRUE;
```

```
    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("set"));
       DerPron := "set";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"7");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 7 ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END ;

8 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("weet"));
       DerPron := "weet";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"8");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 8 ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END ;

9 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("nuf"));
       DerPron := "nuf";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"9");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
```

```
      Trace := "Il appuie sur 9 ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      END;
      END ;

10 :  Trace := "Il appuie sur Point ";

      IF (CompChiffre <=9) THEN
      CompChiffre := CompChiffre + 1;
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
      END;
      IF ActivPoint = FALSE
      THEN
         IF (TableauParametres[1] = 2 ) THEN
          Res := SayAndReturn (ADR("point"));
          DerPron := "point";
         END;
         wMove (Won^,1,1);
         wClrEndLine (Won^);
         ConcatString (StrSomme,".");
         wMove (Won^,2,1);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,12,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);
         ActivPoint := TRUE;
      END;
      END ;

11 :  Trace := "Il appuie sur Virgule ";

      IF (CompChiffre <=9) THEN
      CompChiffre := CompChiffre + 1;
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
      END;
      IF ActivPoint = FALSE
      THEN
         IF (TableauParametres[1] = 2 ) THEN
          Res := SayAndReturn (ADR("virgule"));
          DerPron := "virgule";
         END;
         wMove (Won^,1,1);
         wClrEndLine (Won^);
         ConcatString (StrSomme,",");
         wMove (Won^,2,1);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,12,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);
         ActivPoint := TRUE;
      END;
      END ;

12 :  Trace := "Il appuie sur OK  ";
      Reel (StrSomme,OKS);
      IF OKS THEN Somme := ConvStringToReal(StrSomme); END;
      Snouvede := Sactuel-Somme;
      Snouve := Sactuel+Somme;
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
```

```
      END;
      BeginGadgetList ();
      GadgetTypeReq := TRUE;
      GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                       RelVerify});
      AddGadgetTextButton (10,55, ADR("OUI"));
      AddGadgetTextButton (140,55, ADR("NON"));
      AddGadgetTextButton (10,20, ADR("EST-CE BIEN JUSTE ?"));
      GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                 GadgetMutualExcludeSet {});
      GlFin := EndGadgetList ();
      InitRequester (ReqFin);
      WITH ReqFin DO
             OlderRequest := NIL;
             LeftEdge := 55;
             TopEdge := 88;
             Width := 180;
             Height := 80;
             ReqGadget := GlFin;
             ReqBorder := NIL;
             ReqText := NIL;
             Flags := RequesterFlagsSet {};
             BackFill := BYTE(10);
             ReqLayer := NIL;
             ImageBMap := NIL;
      END;
      IF Request (ReqFin, Win^) THEN
         IF (TableauParametres[1] = 2) THEN
            Res := SayAndReturn (ADR("avay voo feeny ?"));
         END;
         DoneFin := FALSE;
         WHILE NOT DoneFin DO
             Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
             LOOP
                MsgFin := GetMsg (Win^.UserPort^);
                IF (MsgFin = NIL) THEN EXIT; END;
                ProcIMsg (WpFin, MsgFin);
             END;
         END;
      END;

      FreeGadgetList (GlFin^);
      Delay(25);
      IF Fini THEN Done := TRUE END;

13 :    IF TableauChaine.Indice < 499
        THEN TableauChaine.Tab[TableauChaine.Indice] :=28;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
             TableauChaine.Tab[TableauChaine.Indice] :=26;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
```

```
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
      TableauChaine.Indice := TableauChaine.Indice + 1;
END;
Trace := "Il appuie sur RECOMMENCER ";
IF TableauTraces.Indice <= 499 THEN
      CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
      TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                 RelVerify});
AddGadgetTextButton (10,55, ADR("OUI"));
AddGadgetTextButton (180,55, ADR("NON"));
AddGadgetTextButton (4,20, ADR("VOULEZ-VOUS RECOMMENCER ?"));
GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
          GadgetMutualExcludeSet {});
GlRec := EndGadgetList ();
InitRequester (ReqRec);
WITH ReqRec DO
      OlderRequest := NIL;
      LeftEdge := 35;
      TopEdge := 88;
      Width := 220;
      Height := 80;
      ReqGadget := GlRec;
      ReqBorder := NIL;
      ReqText := NIL;
      Flags := RequesterFlagsSet {};
      BackFill := BYTE(10);
      ReqLayer := NIL;
      ImageBMap := NIL;
END;
IF Request (ReqRec, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("Voolay voo ra comohsay ?"));
   END;
   DoneRec := FALSE;
   WHILE NOT DoneRec DO
     Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
     LOOP
        MsgRec := GetMsg (Win^.UserPort^);
        IF (MsgRec = NIL) THEN EXIT; END;
        ProcIMsg (WpRec, MsgRec);
      END;
 END;
END;
FreeGadgetList (GlRec^);
Delay(25);
IF Recom THEN
      CompChiffre :=0;
      wMove (Won^,1,1);
      wClrEndLine (Won^);
      wMove (Won^,2,1);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,12,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      ES := "";
      Der.Last := 0.;
```

```
                Der.X := 150;
                AuMoinsUn := FALSE;
                ActivEff := FALSE;
          (*        For := Decimal;    *)
                CopyString (DerPron,PPron);
             END |

       14 :  IF (TableauParametres[1] = 2) THEN
                Res := SayAndReturn (ADR(DerPron));
             END;
  END;
CloseSpeech ();
END;
END GadgetHandlerCA;

(*****************************************************************************)
(*                              SaisieSommeCalculette                        *)
(*****************************************************************************)

PROCEDURE CorrPMCA (VAR Phrase : ARRAY OF CHAR;
                    VAR PhrasePron : ARRAY OF CHAR );



BEGIN
    IF TableauChaine.Indice <= 499
    THEN TableauChaine.Tab[TableauChaine.Indice] :=25;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
        TableauChaine.Indice := TableauChaine.Indice + 1;
    END;

    For := Decimal;
    CopyString (PPron,PhrasePron);
    ES := "";
    QR := FALSE;
    Somme := Sinit;
    SommeMax := Max;
    Zero := 0.;
    Der.Last := 0.;
    Der.X := 150;
    AuMoinsUn := FALSE;
    ActivEff := FALSE;
    ActivPoint := FALSE;
    CopyString (StrSomme,"");
    CompChiffre := 0;
    ConvRealToString (Somme,StrSomme,2,For);

    WITH Wp DO
      procGadgetUp := GadgetHandlerCA;
    END;
    WITH WpCor DO
      procGadgetUp := GadgetHandlerCorCA;
    END;
    WITH WpFin DO
      procGadgetUp := GadgetHandlerFinCA;
```

```
END;
WITH WpRec DO
  procGadgetUp := GadgetHandlerRecCA;
END;
Scr := CreateScreen (320,256,5,NIL);
IF (Scr # NIL) THEN
  cmap[00] := 0000H;
  cmap[01] := 0FFFH;
  cmap[02] := 0E00H;
  cmap[03] := 0A00H;
  cmap[04] := 0D80H;
  cmap[05] := 0FE0H;
  cmap[06] := 0FCAH;
  cmap[07] := 0080H;
  cmap[08] := 00B6H;
  cmap[09] := 00DDH;
  cmap[10] := 00AFH;
  cmap[11] := 007CH;
  cmap[12] := 000FH;
  cmap[13] := 070FH;
  cmap[14] := 0C0EH;
  cmap[15] := 0C08H;
  cmap[16] := 0620H;
  cmap[17] := 0E52H;
  cmap[18] := 0A52H;
  cmap[19] := 0FCAH;
  cmap[20] := 0333H;
  cmap[21] := 0444H;
  cmap[22] := 0555H;
  cmap[23] := 0666H;
  cmap[24] := 0777H;
  cmap[25] := 0888H;
  cmap[26] := 0999H;
  cmap[27] := 0AAAH;
  cmap[28] := 0CCCH;
  cmap[29] := 0DDDH;
  cmap[30] := 0EEEH;
  cmap[31] := 0FFFH;
  LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

  ImgPtr := FindImageTable(44333333H, ImgCount);
  IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
    Indice := 0;
    WHILE (Indice <= ImgCount-1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr^[Indice].Width;
        Height      := ImgPtr^[Indice].Height;
        Depth       := ImgPtr^[Indice].Depth;
        ImageData   := ImgPtr^[Indice].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice := Indice + 1;
    END;

 ImgPtr2 := FindImageTable(44444444H, ImgCount2);
  IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
   WHILE (Indice-ImgCount <= ImgCount2-1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr2^[Indice-ImgCount].Width;
        Height      := ImgPtr2^[Indice-ImgCount].Height;
```

```
        Depth       := ImgPtr2^[Indice-ImgCount].Depth;
        ImageData   := ImgPtr2^[Indice-ImgCount].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice := Indice + 1;
    END;
 ImgPtr3 := FindImageTable(44777777H, ImgCount3);
 IF (ImgPtr3 # NIL) AND (ImgCount3 # 0) THEN
  WHILE (Indice-ImgCount-ImgCount2 <= ImgCount3-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr3^[Indice-ImgCount-ImgCount2].Width;
       Height      := ImgPtr3^[Indice-ImgCount-ImgCount2].Height;
       Depth       := ImgPtr3^[Indice-ImgCount-ImgCount2].Depth;
       ImageData   := ImgPtr3^[Indice-ImgCount-ImgCount2].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     Indice := Indice + 1;
   END;
ImgPtr4 := FindImageTable(66666666H, ImgCount4);
 IF (ImgPtr4 # NIL) AND (ImgCount4 # 0) THEN
  list6 := 3;
  WHILE (list6 <= ImgCount4-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr4^[list6].Width;
       Height      := ImgPtr4^[list6].Height;
       Depth       := ImgPtr4^[list6].Depth;
       ImageData   := ImgPtr4^[list6].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list6 := list6 + 1;
     Indice := Indice + 1;
   END;
ImgPtr5 := FindImageTable(11111114H, ImgCount5);
 IF (ImgPtr5 # NIL) AND (ImgCount5 # 0) THEN
  list7 := 3;
  WHILE (list7 <= ImgCount5-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr5^[list7].Width;
       Height      := ImgPtr5^[list7].Height;
       Depth       := ImgPtr5^[list7].Depth;
       ImageData   := ImgPtr5^[list7].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list7 := list7 + 1;
     Indice := Indice + 1;
   END;
    BeginGadgetList();
    AddGadgetImageButton (115,147,TabImages[0]);
    AddGadgetImageButton (115,90,TabImages[1]);
    AddGadgetImageButton (135,90,TabImages[2]);
    AddGadgetImageButton (155,90,TabImages[3]);
    AddGadgetImageButton (115,109,TabImages[4]);
```

```
AddGadgetImageButton (155,109,TabImages[5]);
AddGadgetImageButton (155,109,TabImages[6]);
AddGadgetImageButton (115,128,TabImages[7]);
AddGadgetImageButton (135,128,TabImages[8]);
AddGadgetImageButton (155,128,TabImages[9]);
AddGadgetImageButton (135,147,TabImages[10]);
AddGadgetImageButton (155,147,TabImages[28]);
AddGadgetImageButton (135,220,TabImages[21]);
AddGadgetImageButton (65,220,TabImages[48]);

IF TableauParametres[1] = 2 THEN
        AddGadgetImageButton (5,210,TabImages[47])
END;
Gl := EndGadgetList();
IF (Gl # NIL) THEN
   Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,Gl,Scr,NIL);
   Rp := Win^.RPort;
   SetAPen (Rp^,5);
   Move (Rp^,0,0);
   Draw (Rp^,0,255);
   Draw (Rp^,319,255);
   Draw (Rp^,319,0);
   Draw (Rp^,0,0);
   SetAPen (Rp^,1);
   Won := CreateWindow (85,70,130,12,WIDCMP,WFlags,NIL,Scr,NIL);
   Wun := CreateWindow (50,15,264,12,WIDCMP,WFlags2,NIL,Scr,NIL);
   IF (Win # NIL) AND (Won # NIL)  THEN
   IF CreateConsole (Wun^) THEN
      wClrScr (Wun^);
      wSetCursor (Wun^,FALSE);
      wMove (Wun^,1,1);
      PutStr (Wun^,ADR(Phrase));
      DeleteConsole (Wun^);
   END;
   IF CreateConsole (Won^) THEN
      wClrScr (Won^);
      wMove (Won^,1,1);
      wClrEndLine (Won^);
      wMove (Won^,2,1);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,12,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);

      IF NOT QR THEN
        DrawImage (Win^.RPort^,TabImages[54],0,0);
      END;
      CopyString (DerPron,PhrasePron);

  Done := FALSE;
  BeginGadgetList ();
  GadgetTypeReq := TRUE;
  GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                   RelVerify});
  AddGadgetTextButton (10,55, ADR("OUI"));
  AddGadgetTextButton (180,55, ADR("NON"));
  AddGadgetTextButton (8,20, ADR("VOULEZ-VOUS CORRIGER ?"));
  GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
           GadgetMutualExcludeSet {});
  GlCor := EndGadgetList ();
  InitRequester (ReqCor);
  WITH ReqCor DO
       OlderRequest := NIL;
       LeftEdge := 35;
       TopEdge := 88;
       Width := 220;
```

```
      Height := 80;
      ReqGadget := GlCor;
      ReqBorder := NIL;
      ReqText := NIL;
      Flags := RequesterFlagsSet {};
      BackFill := BYTE(10);
      ReqLayer := NIL;
      ImageBMap := NIL;
END;

IF Request (ReqCor, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
     IF OpenSpeech () THEN
        Res := SayAndReturn (ADR("Voolay voo corrigay ?"));
        CloseSpeech ();
     END;
   END;
   DoneCor := FALSE;
   WHILE NOT DoneCor DO
     Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
     LOOP
        MsgCor := GetMsg (Win^.UserPort^);
        IF (MsgCor = NIL) THEN EXIT; END;
        ProcIMsg (WpCor, MsgCor);
     END;
   END;
END;
FreeGadgetList (GlCor^);
Delay(25);
IF Corri THEN
     CompChiffre :=0;
     Somme := 0. ;
     wMove (Won^,1,1);
     wClrEndLine (Won^);
     wMove (Won^,2,1);
     StrSomme := "";
     PutStr (Won^,ADR(StrSomme));
     wMove (Won^,12,1);
     PutStr (Won^,ADR("Frs"));
     wSetCursor (Won^,FALSE);
     ES := "";
     Der.Last := 0.;
     Der.X := 150;
     AuMoinsUn := FALSE;
     ActivPoint := FALSE;
     ActivEff := FALSE;
     CopyString (DerPron,PPron);
 ELSE Done := TRUE;
 END;

  WHILE (NOT Done) DO
      Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
      LOOP
        Msg := GetMsg(Win^.UserPort^);
        IF (Msg = NIL) THEN EXIT; END;
        ProcIMsg (Wp, Msg);
      END;
  END;
  DeleteConsole (Won^);
  END;

Sinit := Somme;
CloseWindow(Wun^);
CloseWindow(Won^);
CloseWindow(Win^);
END;
```

```
            FreeGadgetList (Gl^);
          END;
        CloseScreen(Scr^);
       END;
      END;
     END;
    END;
  END;
END;

END CorrPMCA;

(* VAR
   p : ARRAY [0..40] OF CHAR;
  pp : ARRAY [0..40] OF CHAR;
   q : BOOLEAN;
   v : REAL;
   e : Str20;

BEGIN
   p := "Combien avez-vous d'argent ?";
   pp := "Cobya ah vay voo darjo ?";
   q := FALSE;
   CorrPMCA (p,pp);  *)

END CorrectionPMCA.
```

```
IMPLEMENTATION MODULE CorrPMBillets;

FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr, BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton,AddGadgetImageButton,GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
       StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40, TabNombre,TableauTraces,
       TRecettes,TDepenses,TabNombreInit,Max,Sactuel,Sinit,Str100,
       TableauParametres,TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM Drawing IMPORT RectFill,SetAPen,Draw,Move;
FROM Text IMPORT Text;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Utilitaires IMPORT ConvPrononcable;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre,DoneRec,Recom : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff,QR : BOOLEAN;
  For : RealToStringFormat;
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Win,Won,Win2,Wun : WindowPtr;
  Icone: ARRAY [0..3] OF Image;
  Commande: ARRAY [0..5] OF Image;
```

```
    ConvRealToString (Somme, StrSomme, 0, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 5000 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
1 : DateStamp(DSR);
    NminEND[IndTemps]:=SHORT (DSR.dsMinute);
    NticksEND[IndTemps] := SHORT (DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb1000 := Nb1000 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 1000.;
    IF Nb1000 <= 7
    THEN
       ree := real (Nb1000);
       ree2 := ree/2.;
       ent := entier (ree2);
       re2 := real (ent);
       IF ree2 = re2
       THEN
         Der.Y := 55;
         Der.X := 188 + ((Nb1000 - 2) * 17);
         Der.X := Der.X + ((Nb1000 - 2) DIV 2);
       ELSE
         Der.Y := 62;
         Der.X := 170 + ((Nb1000 - 1) * 18);
         Der.X := Der.X - ((Nb1000 - 1) DIV 2);
       END;
       Der.Type := "BIL";
       Der.Last := 1000.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,Mini[1],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 1000.
    END;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("myl froh"));
       DerPron := "myl froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 0, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 1000 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
2 : DateStamp(DSR);
    NminEND[IndTemps]:=SHORT (DSR.dsMinute);
    NticksEND[IndTemps] := SHORT (DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb500 := Nb500 + 1;
    SommePrecedente := Somme;
```

```
  Monnaie: ARRAY [0..8] OF Image;
  Mini: ARRAY [0..8] OF Image;
  ImgPtr,ImgPtr6,ImgPtr7,ImgPtr8 : ImageDescTablePtr;
  ImgCount,ImgCount6,ImgCount7,ImgCount8,Indice,list6,list7, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK,GlRec : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK,WpRec : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK,ReqRec : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK,MsgRec : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50,Nb20,Nb5,Nb1 : CARDINAL;
  B100, B50, B20, P5, P1,IRecettes,IDepenses,Fait,I,ent : INTEGER;
  Somme, SommePrecedente, SommeMax ,SA,SAP: REAL;
  StrSomme,ES,fonc,Maxaf : Str20;
  Res,Quit : BOOLEAN;
  DerPron,PPron : Str100;
  In : CARDINAL;
  ActMinute,ActTick,ActTick2 : INTEGER;
  ree,ree2,re2 : REAL;
  DSR : DateStampRecord;
  Rp : RastPortPtr;

PROCEDURE GadgetHandlerEffBB (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Efface := TRUE ; DoneEff := TRUE;
           Trace := "Il confirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END; |
     1 : Efface := FALSE; DoneEff := TRUE;
           Trace := "Il infirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END; |
     2 :
  END;
END GadgetHandlerEffBB;


PROCEDURE GadgetHandlerFinBB (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Fini := TRUE ; DoneFin := TRUE;
           IF TableauChaine.Indice < 499
           THEN TableauChaine.Tab[TableauChaine.Indice] :=29;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;                        DateStamp(DSR);

              TableauChaine.Indice := TableauChaine.Indice + 1;
```

```
                TableauChaine.Tab[TableauChaine.Indice] :=7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il confirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; ¦

    1 : Fini := FALSE; DoneFin := TRUE;
            IF TableauChaine.Indice < 499
            THEN TableauChaine.Tab[TableauChaine.Indice] :=29;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
                TableauChaine.Tab[TableauChaine.Indice] :=26;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; ¦
    2 :
  END;
END GadgetHandlerFinBB;

PROCEDURE GadgetHandlerRecBB (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget)
BEGIN
  CASE Gad.GadgetID OF
     0 : Recom := TRUE ; DoneRec := TRUE;
           Trace := "Il confirme la demande ";
```

```
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
      1 : Recom := FALSE; DoneRec := TRUE;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
      2 :
   END;
END GadgetHandlerRecBB;

PROCEDURE GadgetHandlerBB (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
    SomAct : REAL;
BEGIN
   IF OpenSpeech () THEN
   CASE Gad.GadgetID OF
      0 : ActivEff := TRUE;
          Nb5000 := Nb5000 + 1;
          SommePrecedente := Somme;
          Somme := Somme + 5000.;
          IF Nb5000 <= 7
          THEN
             ree := real (Nb5000);
             ree2 := ree/2.;
             ent := entier (ree2);
             re2 := real (ent);
             IF ree2 = re2
             THEN
               Der.Y := 26;
               Der.X := 189 + ((Nb5000 - 2) * 18);
               Der.X := Der.X + ((Nb5000 - 2) DIV 2);
             ELSE
               Der.Y := 33;
               Der.X := 170 + ((Nb5000 - 1) * 19);
               Der.X := Der.X - ((Nb5000 - 1) DIV 2);
             END;
             Der.Type := "BIL";
             Der.Last := 5000.;
             AuMoinsUn := TRUE;
             Delay (25);
             DrawImage (Win^.RPort^,Mini[0],Der.X,Der.Y)
          ELSE
             AuMoinsUn := FALSE;
             Der.Last := 5000.
          END;
          IF (TableauParametres[1] = 2) THEN
             Res := SayAndReturn (ADR("senk myl froh"));
             DerPron := "senk myl froh";
          END;
          wMove (Won^,7,1);
          wClrEndLine (Won^);
          ConvRealToString (Somme, StrSomme, 0, For);
          PutStr (Won^,ADR(StrSomme));
          wMove (Won^,15,1);
          PutStr (Won^,ADR("Frs"));
          wSetCursor (Won^,FALSE);
          Trace := "Il appuie sur 5000 Frs ";
          IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
          END |
```

```
1 : ActivEff := TRUE;
    Nb1000 := Nb1000 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 1000.;
    IF Nb1000 <= 7
    THEN
       ree := real (Nb1000);
       ree2 := ree/2.;
       ent := entier (ree2);
       re2 := real (ent);
       IF ree2 = re2
       THEN
         Der.Y := 55;
         Der.X := 188 + ((Nb1000 - 2) * 17);
         Der.X := Der.X + ((Nb1000 - 2) DIV 2);
       ELSE
         Der.Y := 62;
         Der.X := 170 + ((Nb1000 - 1) * 18);
         Der.X := Der.X - ((Nb1000 - 1) DIV 2);
       END;
       Der.Type := "BIL";
       Der.Last := 1000.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,Mini[1],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 1000.
    END;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("myl froh"));
       DerPron := "myl froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 0, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 1000 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;

2 : ActivEff := TRUE;
    Nb500 := Nb500 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 500.;
    IF Nb500 <= 8
    THEN
       ree := real (Nb500);
       ree2 := ree/2.;
       ent := entier (ree2);
       re2 := real (ent);
       IF ree2 = re2
       THEN
         Der.Y := 84;
         Der.X := 187 + ((Nb500 - 2) * 16);
         Der.X := Der.X + ((Nb500 - 2) DIV 2);
       ELSE
         Der.Y := 91;
         Der.X := 170 + ((Nb500 - 1) * 17);
         Der.X := Der.X - ((Nb500 - 1) DIV 2);
```

```
      END;
      Der.Type := "BIL";
      Der.Last := 500.;
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,Mini[2],Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 500.
   END;
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("senk soh froh"));
      DerPron := "senk san froh";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 0, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);
   Trace := "Il appuie sur 500 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END ;

3 : ActivEff := TRUE;
   Nb100 := Nb100 + 1;
   SommePrecedente := Somme;
   Somme := Somme + 100.;
   IF Nb100 <= 8
   THEN
      ree := real (Nb100);
      ree2 := ree/2.;
      ent := entier (ree2);
      re2 := real (ent);
      IF ree2 = re2
      THEN
        Der.Y := 113;
        Der.X := 186 + ((Nb100 - 2) * 15);
        Der.X := Der.X + ((Nb100 - 2) DIV 2);
      ELSE
        Der.Y := 120;
        Der.X := 170 + ((Nb100 - 1) * 16);
        Der.X := Der.X - ((Nb100 - 1) DIV 2);
      END;
      Der.Type := "BIL";
      Der.Last := 100.;
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,Mini[3],Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 100.
   END;
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("soh froh"));
      DerPron := "soh froh";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 0, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
```

```
        wSetCursor (Won^,FALSE);
        Trace := "Il appuie sur 100 Frs ";
        IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ;

    4 : ActivEff := TRUE;
        Nb50 := Nb50 + 1;
        SommePrecedente := Somme;
        Somme := Somme + 50.;
        IF Nb50 <= 8
        THEN
           Der.X := 170 + ((Nb50 - 1) * 18);
           Der.Y := 145;
           Der.Type := "PA";
           Der.Last := 50.;
           AuMoinsUn := TRUE;
           Delay (25);
           DrawImage (Win^.RPort^,Mini[4],Der.X,Der.Y)
        ELSE
           AuMoinsUn := FALSE;
           Der.Last := 50.
        END;
        IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("sekan't froh"));
           DerPron := "sekan't froh";
        END;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 0, For);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        Trace := "Il appuie sur 50 Frs ";
        IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ;

    5 : ActivEff := TRUE;
        Nb20 := Nb20 + 1;
        SommePrecedente := Somme;
        Somme := Somme + 20.;
        IF Nb20 <= 8
        THEN
           Der.X := 170 + ((Nb20 - 1) * 18);
           Der.Y := 168;
           Der.Type := "PA";
           Der.Last := 20.;
           AuMoinsUn := TRUE;
           Delay (25);
           DrawImage (Win^.RPort^,Mini[5],Der.X,Der.Y)
        ELSE
           AuMoinsUn := FALSE;
           Der.Last := 20.
        END;
        IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("vant froh"));
           DerPron := "vant froh";
        END;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 0, For);
        PutStr (Won^,ADR(StrSomme));
```

```
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        Trace := "Il appuie sur 20 Frs ";
        IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ;

    6 : ActivEff := TRUE;
        Nb5 := Nb5 + 1;
        SommePrecedente := Somme;
        Somme := Somme + 5.;
        IF Nb5 <= 8
        THEN
           Der.X := 170 + ((Nb5 - 1) * 18);
           Der.Y := 194;
           Der.Type := "PA";
           Der.Last := 5.;
           AuMoinsUn := TRUE;
           Delay (25);
           DrawImage (Win^.RPort^,Mini[6],Der.X,Der.Y)
        ELSE
           AuMoinsUn := FALSE;
           Der.Last := 5.
        END;
        IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("senk froh"));
           DerPron := "senk froh";
        END;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 0, For);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        Trace := "Il appuie sur 5 Frs ";
        IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ;

    7 : ActivEff := TRUE;
        Nb1 := Nb1 + 1;
        SommePrecedente := Somme;
        Somme := Somme + 1.;
        IF Nb1 <= 8
        THEN
           Der.X := 170 + ((Nb1 - 1) * 18);
           Der.Y := 218;
           Der.Type := "PA";
           Der.Last := 1.;
           AuMoinsUn := TRUE;
           Delay (25);
           DrawImage (Win^.RPort^,Mini[7],Der.X,Der.Y)
        ELSE
           AuMoinsUn := FALSE;
           Der.Last := 1.
        END;
        IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("an froh"));
           DerPron := "an froh";
        END;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
```

```
      ConvRealToString (Somme, StrSomme, 0, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 1 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;

8 : IF ActivEff THEN
      IF TableauChaine.Indice < 499
      THEN TableauChaine.Tab[TableauChaine.Indice] :=27;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
           TableauChaine.Indice := TableauChaine.Indice + 1;
           TableauChaine.Tab[TableauChaine.Indice] :=26;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
           TableauChaine.Indice := TableauChaine.Indice + 1;
      END;
      Trace := "Il demande d'effacer ";
      IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      BeginGadgetList ();
      GadgetTypeReq := TRUE;
      GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                       RelVerify});
      AddGadgetTextButton (10,34, ADR("OUI"));
      AddGadgetTextButton (140,34, ADR("NON"));
      AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS EFFACER ?"));
      GadgetOpt (GadgetFlagsSet{}, GadgetActivationSet {RelVerify},
                 GadgetMutualExcludeSet {});
      GlEff := EndGadgetList ();
      InitRequester (ReqEff);
      WITH ReqEff DO
             OlderRequest := NIL;
             LeftEdge := 70;
             TopEdge := 22;
             Width := 180;
             Height := 50;
             ReqGadget := GlEff;
             ReqBorder := NIL;
```

```
        ReqText := NIL;
        Flags := RequesterFlagsSet {};
        BackFill := BYTE(10);
        ReqLayer := NIL;
        ImageBMap := NIL;
END;
IF Request (ReqEff, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("vooley voo effassay ?"));
   END;
   DoneEff := FALSE;
   WHILE NOT DoneEff DO
       Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
       LOOP
          MsgEff := GetMsg (Win^.UserPort^);
          IF (MsgEff = NIL) THEN EXIT; END;
          ProcIMsg (WpEff, MsgEff);
       END;
   END;
END;
FreeGadgetList (GlEff^);
Delay(25);
IF Efface THEN
          IF Der.Last = 5000.
          THEN
            Nb5000 := Nb5000 - 1;
          ELSIF Der.Last = 1000.
          THEN
           Nb1000 := Nb1000 - 1;
          ELSIF Der.Last = 500.
          THEN
            Nb500 := Nb500 - 1;
          ELSIF Der.Last = 100.
          THEN
            Nb100 := Nb100 - 1;
          ELSIF Der.Last = 50.
          THEN
            Nb50 := Nb50 - 1;
          ELSIF Der.Last = 20.
          THEN
            Nb20 := Nb20 - 1;
          ELSIF Der.Last = 5.
          THEN
            Nb5 := Nb5 - 1;
          ELSIF Der.Last = 1.
          THEN
            Nb1 := Nb1 - 1;
          END;

  ActivEff := FALSE;
  IF ((CompareString (Der.Type,"BIL") = equal) AND (AuMoinsUn))
  THEN
      SetAPen (Rp^,0);
      IF Der.Last = 500.
      THEN
        RectFill (Rp^,Der.X,Der.Y,Der.X+31,Der.Y+13);
      ELSE
        RectFill (Rp^,Der.X,Der.Y,Der.X+35,Der.Y+13);
      END;
      SetAPen (Rp^,1);
      IF (Der.X # 170) THEN
        IF Der.Last = 5000.
          THEN
            ree := real (Nb5000);
            ree2 := ree/2.;
            ent := entier (ree2);
```

```
            re2 := real (ent);
            IF ree2 = re2
            THEN
            Der.Y := 26;
            Der.X := 189 + ((Nb5000 - 2) * 18);
            Der.X := Der.X + ((Nb5000 - 2) DIV 2);
            ELSE
            Der.Y := 33;
            Der.X := 170 + ((Nb5000 - 1) * 19);
            Der.X := Der.X - ((Nb5000 - 1) DIV 2);
            END;
            DrawImage (Win^.RPort^,Mini[0],Der.X,Der.Y)
          ELSIF Der.Last = 1000.
          THEN
           ree := real (Nb1000);
           ree2 := ree/2.;
           ent := entier (ree2);
           re2 := real (ent);
           IF ree2 = re2
           THEN
             Der.Y := 55;
             Der.X := 188 + ((Nb1000 - 2) * 17);
             Der.X := Der.X + ((Nb1000 - 2) DIV 2);
           ELSE
             Der.Y := 62;
             Der.X := 170 + ((Nb1000 - 1) * 18);
             Der.X := Der.X - ((Nb1000 - 1) DIV 2);
           END;
           DrawImage (Win^.RPort^,Mini[1],Der.X,Der.Y)
          ELSIF Der.Last = 500.
          THEN
            ree := real (Nb500);
            ree2 := ree/2.;
            ent := entier (ree2);
            re2 := real (ent);
            IF ree2 = re2
            THEN
              Der.Y := 84;
              Der.X := 187 + ((Nb500 - 2) * 16);
              Der.X := Der.X + ((Nb500 - 2) DIV 2);
            ELSE
              Der.Y := 91;
              Der.X := 170 + ((Nb500 - 1) * 17);
              Der.X := Der.X - ((Nb500 - 1) DIV 2);
            END;
            DrawImage (Win^.RPort^,Mini[2],Der.X,Der.Y)
          ELSIF Der.Last = 100.
          THEN
            ree := real (Nb100);
            ree2 := ree/2.;
            ent := entier (ree2);
            re2 := real (ent);
            IF ree2 = re2
            THEN
              Der.Y := 113;
              Der.X := 186 + ((Nb100 - 2) * 15);
              Der.X := Der.X + ((Nb100 - 2) DIV 2);
            ELSE
              Der.Y := 120;
              Der.X := 170 + ((Nb100 - 1) * 16);
              Der.X := Der.X - ((Nb100 - 1) DIV 2);
            END;
            DrawImage (Win^.RPort^,Mini[3],Der.X,Der.Y)

        END
      END
```

```
        ELSIF ((CompareString (Der.Type,"PA") = equal) AND (AuMoinsUn))
        THEN SetAPen (Rp^,0);
             RectFill (Rp^,Der.X,Der.Y,Der.X +16,Der.Y+16);
             SetAPen (Rp^,1);

        END;
        SommePrecedente := Somme;
        Somme := Somme - Der.Last;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 0, For);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        DerPron := "";

      END
    END ;

9:  Trace := "Il appuie sur OK  ";
    IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    BeginGadgetList ();
    GadgetTypeReq := TRUE;
    GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                     RelVerify});
    AddGadgetTextButton (10,34, ADR("OUI"));
    AddGadgetTextButton (140,34, ADR("NON"));
    AddGadgetTextButton (8,9, ADR("EST-CE BIEN JUSTE ?"));
    GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
               GadgetMutualExcludeSet {});
    GlFin := EndGadgetList ();
    InitRequester (ReqFin);
    WITH ReqFin DO
           OlderRequest := NIL;
           LeftEdge := 70;
           TopEdge := 22;
           Width := 180;
           Height := 50;
           ReqGadget := GlFin;
           ReqBorder := NIL;
           ReqText := NIL;
           Flags := RequesterFlagsSet {};
           BackFill := BYTE(10);
           ReqLayer := NIL;
           ImageBMap := NIL;
    END;
    IF Request (ReqFin, Win^) THEN
       IF (TableauParametres[1] = 2) THEN
          Res := SayAndReturn (ADR("avay voo feeny ?"));
       END;
       DoneFin := FALSE;
       WHILE NOT DoneFin DO
           Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
           LOOP
              MsgFin := GetMsg (Win^.UserPort^);
              IF (MsgFin = NIL) THEN EXIT; END;
              ProcIMsg (WpFin, MsgFin);
           END;
       END;
    END;

    FreeGadgetList (GlFin^);
```

```
    Delay(25);
    IF Fini THEN Done := TRUE END;

10: IF TableauChaine.Indice < 499
    THEN TableauChaine.Tab[TableauChaine.Indice] :=28;
              DateStamp(DSR);
              ActMinute := SHORT(DSR.dsMinute);
              ActTick := SHORT(DSR.dsTick);
              ActTick2 := ActTick;
              IF ActMinute > AncMinute
              THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
              END;
              ent := ActTick - AncTick;
              ree := real(ent);
              TableauTemps [TableauChaine.Indice] := entier(ree/50.);
              AncMinute := ActMinute;
              AncTick := ActTick2;
         TableauChaine.Indice := TableauChaine.Indice + 1;
         TableauChaine.Tab[TableauChaine.Indice] :=26;
              DateStamp(DSR);
              ActMinute := SHORT(DSR.dsMinute);
              ActTick := SHORT(DSR.dsTick);
              ActTick2 := ActTick;
              IF ActMinute > AncMinute
              THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
              END;
              ent := ActTick - AncTick;
              ree := real(ent);
              TableauTemps [TableauChaine.Indice] := entier(ree/50.);
              AncMinute := ActMinute;
              AncTick := ActTick2;
         TableauChaine.Indice := TableauChaine.Indice + 1;
    END;
    Trace := "Il appuie sur RECOMMENCER ";
    IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    BeginGadgetList ();
    GadgetTypeReq := TRUE;
    GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                     RelVerify});
    AddGadgetTextButton (10,34, ADR("OUI"));
    AddGadgetTextButton (180,34, ADR("NON"));
    AddGadgetTextButton (4,9, ADR("VOULEZ-VOUS RECOMMENCER ?"));
    GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
              GadgetMutualExcludeSet {});
    GlRec := EndGadgetList ();
    InitRequester (ReqRec);
    WITH ReqRec DO
         OlderRequest := NIL;
         LeftEdge := 50;
         TopEdge := 22;
         Width := 220;
         Height := 50;
         ReqGadget := GlRec;
         ReqBorder := NIL;
         ReqText := NIL;
         Flags := RequesterFlagsSet {};
         BackFill := BYTE(10);
         ReqLayer := NIL;
         ImageBMap := NIL;
    END;
    IF Request (ReqRec, Win^) THEN
       IF (TableauParametres[1] = 2) THEN
         Res := SayAndReturn (ADR("Voolay voo ra comohsay ?"));
```

```
            END;
            DoneRec := FALSE;
            WHILE NOT DoneRec DO
              Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
              LOOP
                 MsgRec := GetMsg (Win^.UserPort^);
                 IF (MsgRec = NIL) THEN EXIT; END;
                 ProcIMsg (WpRec, MsgRec);
              END;
          END;
        END;
        FreeGadgetList (GlRec^);
        Delay(25);
        IF Recom THEN
             SetAPen (Rp^,0);
             RectFill (Rp^,170,22,317,236);
             SetAPen (Rp^,1);
             Somme := 0.;
             wMove (Won^,7,1);
             wClrEndLine (Won^);
             ConvRealToString (Somme, StrSomme, 2, For);
             PutStr (Won^,ADR(StrSomme));
             wMove (Won^,15,1);
             PutStr (Won^,ADR("Frs"));
             wSetCursor (Won^,FALSE);
             ES := "";
             SommeMax := Max;
             Nb100 := 0;
             Nb50 := 0;
             Nb20 := 0;
             Nb1000 := 0;
             Nb500 := 0;
             Nb5000 := 0;
             Nb5 := 0;
             Nb1 := 0;
             Der.Last := 0.;
             Der.X := 150;
             AuMoinsUn := FALSE;
             ActivEff := FALSE;
             For := Decimal;
             CopyString (DerPron,PPron);

        END;

      11:  IF (TableauParametres[1] = 2) THEN
             Res := SayAndReturn (ADR(DerPron));
           END;
  END;
CloseSpeech ();
END;
END GadgetHandlerBB;

(*************************************************************************
(*                          Correction Porte Monnaie                     *
(*************************************************************************

PROCEDURE CorrPMBI (VAR Phrase : ARRAY OF CHAR;
                    VAR PhrasePron : ARRAY OF CHAR );



BEGIN
    IF TableauChaine.Indice <= 499
    THEN TableauChaine.Tab[TableauChaine.Indice] :=26;
                DateStamp(DSR);
```

```
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
      TableauChaine.Indice := TableauChaine.Indice + 1;
END;

CopyString (PPron,PhrasePron);
QR := FALSE;
Somme := 0.;
SommeMax := Max;
Nb100 := 0;
Nb50 := 0;
Nb20 := 0;
Nb1000 := 0;
Nb500 := 0;
Nb5000 := 0;
Nb5 := 0;
Nb1 := 0;
Der.Last := 0.;
Der.X := 150;
AuMoinsUn := FALSE;
ActivEff := FALSE;
For := Decimal;

WITH Wp DO
  procGadgetUp := GadgetHandlerBB;
END;
WITH WpEff DO
  procGadgetUp := GadgetHandlerEffBB;
END;
WITH WpFin DO
  procGadgetUp := GadgetHandlerFinBB;
END;
WITH WpRec DO
  procGadgetUp := GadgetHandlerRecBB;
END;
Scr := CreateScreen (320,256,5,NIL);
IF (Scr # NIL) THEN
  cmap[00] := 0000H;
  cmap[01] := 0FFFH;
  cmap[02] := 0E00H;
  cmap[03] := 0A00H;
  cmap[04] := 0D80H;
  cmap[05] := 0FE0H;
  cmap[06] := 0FCAH;
  cmap[07] := 0080H;
  cmap[08] := 00B6H;
  cmap[09] := 00DDH;
  cmap[10] := 00AFH;
  cmap[11] := 007CH;
  cmap[12] := 000FH;
  cmap[13] := 070FH;
  cmap[14] := 0C0EH;
  cmap[15] := 0C08H;
  cmap[16] := 0620H;
  cmap[17] := 0E52H;
  cmap[18] := 0A52H;
  cmap[19] := 0FCAH;
```

```
cmap[20] := 0333H;
cmap[21] := 0444H;
cmap[22] := 0555H;
cmap[23] := 0666H;
cmap[24] := 07D8H;
cmap[25] := 0C97H;
cmap[26] := 09CFH;
cmap[27] := 0D9EH;
cmap[28] := 0EB0H;
cmap[29] := 0DDDH;
cmap[30] := 0EC7H;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

ImgPtr := FindImageTable(15111111H, ImgCount);
IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
  Indice := 0;
  WHILE (Indice <= ImgCount-1) DO
    WITH Mini[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr^[Indice].Width;
      Height      := ImgPtr^[Indice].Height;
      Depth       := ImgPtr^[Indice].Depth;
      ImageData   := ImgPtr^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

ImgPtr6 := FindImageTable(12111111H, ImgCount6);
IF (ImgPtr6 # NIL) AND (ImgCount6 # 0) THEN
  Indice := 0;
  WHILE (Indice <= ImgCount6 - 1) DO
    WITH Monnaie[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr6^[Indice].Width;
      Height      := ImgPtr6^[Indice].Height;
      Depth       := ImgPtr6^[Indice].Depth;
      ImageData   := ImgPtr6^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

ImgPtr7 := FindImageTable(13111111H, ImgCount7);
IF (ImgPtr7 # NIL) AND (ImgCount7 # 0) THEN
  Indice := 0;
  WHILE (Indice <= ImgCount7 -1) DO
    WITH Commande [Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr7^[Indice].Width;
      Height      := ImgPtr7^[Indice].Height;
      Depth       := ImgPtr7^[Indice].Depth;
      ImageData   := ImgPtr7^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
```

```
  END;

ImgPtr8 := FindImageTable(14111111H, ImgCount8);
IF (ImgPtr8 # NIL) AND (ImgCount8 # 0) THEN
  Indice := 0;
  WHILE (Indice <= ImgCount8 -1) DO
    WITH Icone[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr8^[Indice].Width;
      Height      := ImgPtr8^[Indice].Height;
      Depth       := ImgPtr8^[Indice].Depth;
      ImageData   := ImgPtr8^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

   BeginGadgetList();
   AddGadgetImageButton (10,59,Monnaie[0]);
   AddGadgetImageButton (12,99,Monnaie[1]);
   AddGadgetImageButton (14,139,Monnaie[2]);
   AddGadgetImageButton (16,179,Monnaie[3]);
   AddGadgetImageButton (93,60,Monnaie[4]);
   AddGadgetImageButton (90,97,Monnaie[5]);
   AddGadgetImageButton (92,139,Monnaie[6]);
   AddGadgetImageButton (96,182,Monnaie[7]);
   AddGadgetImageButton (3,223,Commande[0]);
   AddGadgetImageButton (125,223,Commande[2]);
   AddGadgetImageButton (65,223,Commande[1]);
   IF TableauParametres[1] = 2 THEN
          AddGadgetImageButton (136,175,Commande[4])
   END;
   Gl := EndGadgetList();
   IF (Gl # NIL) THEN
      Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,Gl,Scr,NIL);
      Won := CreateWindow (167,239,151,15,WIDCMP,WFlags2,NIL,Scr,NIL);
      Win2 := CreateWindow (45,5,273,15,WIDCMP,WFlags2,NIL,Scr,NIL);
      IF (Win # NIL) AND (Won # NIL) AND (Win2 # NIL) THEN
       IF CreateConsole (Win2^) THEN
         wClrScr (Win2^);
         PutStr (Win2^,ADR(Phrase));
         wSetCursor (Win2^,FALSE);
       IF CreateConsole (Won^) THEN
         wClrScr (Won^);
         PutStr (Won^,ADR("Total"));
         wMove (Won^,15,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);

         Rp := Win^.RPort;
         DrawImage (Win^.RPort^,Icone[0],0,0);
         SetAPen (Rp^,5);
         Move (Rp^,0,0);
         Draw (Rp^,0,255);
         Draw (Rp^,319,255);
         Draw (Rp^,319,0);
         Draw (Rp^,0,0);
         Move (Rp^,1,1);
         Draw (Rp^,1,254);
         Draw (Rp^,318,254);
         Draw (Rp^,318,1);
         Draw (Rp^,1,1);
         Move (Rp^,165,20);
```

```
      Draw (Rp^,165,255);
      Move (Rp^,166,20);
      Draw (Rp^,166,255);
      Move (Rp^,43,20);
      Draw (Rp^,319,20);
      Move (Rp^,43,21);
      Draw (Rp^,319,21);
      Move (Rp^,166,237);
      Draw (Rp^,319,237);
      Move (Rp^,166,238);
      Draw (Rp^,319,238);

      IF TableauParametres[1] = 2 THEN
         IF OpenSpeech () THEN
               Res := SayAndReturn (ADR(PhrasePron));
               CopyString (DerPron,PhrasePron);
               CloseSpeech ()
         END;
      END;
      SA := Sactuel;

      In := 1;
      WHILE (In <= TabNombreInit [0]) DO

        Nb5000 := Nb5000 + 1;
        SommePrecedente := Somme;
        Somme := Somme + 5000.;

IF Nb5000 <= 7
THEN
   ree := real (Nb5000);
   ree2 := ree/2.;
   ent := entier (ree2);
   re2 := real (ent);
   IF ree2 = re2
   THEN
     Der.Y := 26;
     Der.X := 189 + ((Nb5000 - 2) * 18);
     Der.X := Der.X + ((Nb5000 - 2) DIV 2);
   ELSE
     Der.Y := 33;
     Der.X := 170 + ((Nb5000 - 1) * 19);
     Der.X := Der.X - ((Nb5000 - 1) DIV 2);
   END;
   Der.Type := "BIL";
   Der.Last := 5000.;
   AuMoinsUn := TRUE;
   DrawImage (Win^.RPort^,Mini[0],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 5000.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 0, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

    In := In+1;

      END;

      In :=1;
      WHILE (In <= TabNombreInit [1]) DO
```

```
Nb1000 := Nb1000 + 1;
SommePrecedente := Somme;
Somme := Somme + 1000.;
IF Nb1000 <= 7
THEN
   ree := real (Nb1000);
   ree2 := ree/2.;
   ent := entier (ree2);
   re2 := real (ent);
   IF ree2 = re2
   THEN
     Der.Y := 55;
     Der.X := 188 + ((Nb1000 - 2) * 17);
     Der.X := Der.X + ((Nb1000 - 2) DIV 2);
   ELSE
     Der.Y := 62;
     Der.X := 170 + ((Nb1000 - 1) * 18);
     Der.X := Der.X - ((Nb1000 - 1) DIV 2);
   END;
   Der.Type := "BIL";
   Der.Last := 1000.;
   AuMoinsUn := TRUE;
   DrawImage (Win^.RPort^,Mini[1],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 1000.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 0, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

    In := In+1;
      END;
      In :=1;
      WHILE (In <= TabNombreInit [2]) DO

Nb500 := Nb500 + 1;
SommePrecedente := Somme;
Somme := Somme + 500.;
IF Nb500 <= 8
THEN
   ree := real (Nb500);
   ree2 := ree/2.;
   ent := entier (ree2);
   re2 := real (ent);
   IF ree2 = re2
   THEN
     Der.Y := 84;
     Der.X := 187 + ((Nb500 - 2) * 16);
     Der.X := Der.X + ((Nb500 - 2) DIV 2);
   ELSE
     Der.Y := 91;
     Der.X := 170 + ((Nb500 - 1) * 17);
     Der.X := Der.X - ((Nb500 - 1) DIV 2);
   END;
   Der.Type := "BIL";
   Der.Last := 500.;
   AuMoinsUn := TRUE;
   DrawImage (Win^.RPort^,Mini[2],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
```

```
   Der.Last := 500.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 0, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

    In := In+1;
      END;
      In :=1;
      WHILE (In <= TabNombreInit [3]) DO

Nb100 := Nb100 + 1;
SommePrecedente := Somme;
Somme := Somme + 100.;
IF Nb100 <= 8
THEN
   ree := real (Nb100);
   ree2 := ree/2.;
   ent := entier (ree2);
   re2 := real (ent);
   IF ree2 = re2
   THEN
     Der.Y := 113;
     Der.X := 186 + ((Nb100 - 2) * 15);
     Der.X := Der.X + ((Nb100 - 2) DIV 2);
   ELSE
     Der.Y := 120;
     Der.X := 170 + ((Nb100 - 1) * 16);
     Der.X := Der.X - ((Nb100 - 1) DIV 2);
   END;
   Der.Type := "BIL";
   Der.Last := 100.;
   AuMoinsUn := TRUE;
   DrawImage (Win^.RPort^,Mini[3],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 100.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 0, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

    In := In+1;
      END;

      In :=1;
      WHILE (In <= TabNombreInit [5]) DO

Nb50 := Nb50 + 1;
SommePrecedente := Somme;
Somme := Somme + 50.;
IF Nb50 <= 8
THEN
   Der.X := 170 + ((Nb50 - 1) * 18);
   Der.Y := 145;
   Der.Type := "PA";
   Der.Last := 50.;
   AuMoinsUn := TRUE;
```

```
   DrawImage (Win^.RPort^,Mini[4],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 50.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 0, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

    In := In+1;
      END;
      In :=1;
      WHILE (In <= TabNombreInit [6]) DO

Nb20 := Nb20 + 1;
SommePrecedente := Somme;
Somme := Somme + 20.;
IF Nb20 <= 8
THEN
   Der.X := 170 + ((Nb20 - 1) * 18);
   Der.Y := 168;
   Der.Type := "PA";
   Der.Last := 20.;
   AuMoinsUn := TRUE;
   DrawImage (Win^.RPort^,Mini [5],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 20.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 0, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

    In := In+1;
      END;
      In :=1;
      WHILE (In <= TabNombreInit [7]) DO

Nb5 := Nb5 + 1;
SommePrecedente := Somme;
Somme := Somme + 5.;
IF Nb5 <= 8
THEN
   Der.X := 170 + ((Nb5 - 1) * 18);
   Der.Y := 194;
   Der.Type := "PA";
   Der.Last := 5.;
   AuMoinsUn := TRUE;
   DrawImage (Win^.RPort^,Mini[6],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 5.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 0, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
```

```
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);

            In :=In+1;
              END;
              In :=1;
              WHILE (In <= TabNombreInit [8]) DO

        Nb1 := Nb1 + 1;
        SommePrecedente := Somme;
        Somme := Somme + 1.;
        IF Nb1 <= 8
        THEN
           Der.X := 170 + ((Nb1 - 1) * 18);
           Der.Y := 218;
           Der.Type := "PA";
           Der.Last := 1.;
           AuMoinsUn := TRUE;
           DrawImage (Win^.RPort^,Mini[7],Der.X,Der.Y)
        ELSE
           AuMoinsUn := FALSE;
           Der.Last := 1.
        END;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 0, For);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);

            In := In+1;
              END;

              Done := FALSE;
              WHILE (NOT Done) DO
                Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                LOOP
                  Msg := GetMsg(Win^.UserPort^);
                  IF (Msg = NIL) THEN EXIT; END;
                  ProcIMsg (Wp, Msg);
                END;
              END;
              DeleteConsole (Won^);
             END;
             DeleteConsole (Win2^);
            END;
          Sinit := Somme;
          CloseWindow(Won^);
          CloseWindow(Win2^);
          CloseWindow(Win^);
          END;
          FreeGadgetList (Gl^);
        END;
       CloseScreen(Scr^);
      END;
     END;
    END;
  END;
END;

TabNombreInit[0] := Nb5000;
TabNombreInit[1] := Nb1000;
TabNombreInit[2] := Nb500;
TabNombreInit[3] := Nb100;
TabNombreInit[5] := Nb50;
```

```
TabNombreInit[6] := Nb20;
TabNombreInit[7] := Nb5;
TabNombreInit[8] := Nb1;

END CorrPMBI;

(*  VAR p,pp : ARRAY [0..40] OF CHAR;

BEGIN
    p := "Combien avez-vous d'argent ?";
    pp := "Cobya ah vay voo darjo ?";
    CorrPMBI (p,pp);  *)

END CorrPMBillets.
```

```
IMPLEMENTATION MODULE CorrectionPMCL;

FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, CloseWindow, CloseScreen, ActivateWindow,Image,DrawImage,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, GadgetTypeReq, GlobalGadgetOpt,GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM InOut IMPORT WriteString;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM Strings IMPORT  CopyString, Relation,ConcatString,StringLength,
        CompareString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100, TabNombre,
        TableauTraces,TRecettes,TDepenses,TabNombreInit,TableauParametres,
        Sinit,Sactuel,Max,TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast, RastPortPtr;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};


VAR
  Nombre : Str20;
  Done,OKNombre,Efface,Res,QR : BOOLEAN;
  Scr : ScreenPtr;
  Win,Win2,Won,Wun,Wun2 : WindowPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  Gl : GadgetPtr;
  Wp : WindowProc;
  Req : Requester;
  Sig : SignalSet;
  Msg : IntuiMessagePtr;
  ES,Maxaf : Str20;
  PPron : Str100;
  ValeurMax,ValeurPrec,Val,SA,SAP : REAL;
  ImageRec,ImageDep : Image;
  ImgPtr,ImgPtr2 : ImageDescTablePtr;
  ImgCount,ImgCount2,Indice,Indlist : CARDINAL;
  TabImages : ARRAY [0..59] OF Image;
  PREM : BOOLEAN;
  StVal : Str20;
  Deb : BOOLEAN;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree : REAL;
  DSR : DateStampRecord;
  Rp : RastPortPtr;
```

```
PROCEDURE GadgetHandlerSSN (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

VAR OK : BOOLEAN;
    V : REAL;
BEGIN

  IF Gad.GadgetID = 0 THEN Done := TRUE; Efface := FALSE;
               IF TableauChaine.Indice <= 499
               THEN IF Deb = TRUE
                    THEN TableauChaine.Tab[TableauChaine.Indice] :=7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    ELSE TableauChaine.Tab[TableauChaine.Indice] :=29;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                         TableauChaine.Tab[TableauChaine.Indice] := 7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    END;
               END;
               Deb := FALSE;
               IF TableauTraces.Indice <= 499 THEN
                  CopyString (TableauTraces.Tab [TableauTraces.Indice],
                              "Il appuie sur OK");
                  TableauTraces.Indice := TableauTraces.Indice + 1;
               END;
  ELSIF Gad.GadgetID = 1
                     THEN  Done := TRUE;
               IF TableauChaine.Indice <= 499
               THEN IF Deb = TRUE
                    THEN TableauChaine.Tab[TableauChaine.Indice] :=26;
                    DateStamp(DSR);
```

```
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    ELSE TableauChaine.Tab[TableauChaine.Indice] :=28;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                         TableauChaine.Tab[TableauChaine.Indice] := 26;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    END;
               END;
               Deb := FALSE;
                         IF TableauTraces.Indice <= 499 THEN
                         CopyString (TableauTraces.Tab [TableauTraces.Indice
                                   "Il appuie sur RECOMMENCER");
                         TableauTraces.Indice := TableauTraces.Indice + 1;
                         END;
                    ELSE Done := TRUE;
                       ES := "quitter";
                       Efface := FALSE;
                         IF TableauTraces.Indice <= 499 THEN
                         CopyString (TableauTraces.Tab [TableauTraces.Indice
                                   "Il appuie sur QUITTER");
                         TableauTraces.Indice := TableauTraces.Indice + 1;
                         END;
  END;
END GadgetHandlerSSN;


PROCEDURE Reel (VAR St : ARRAY OF CHAR ;  VAR OK : BOOLEAN);
  VAR Indice : CARDINAL;
      Premier : BOOLEAN;
  BEGIN
    Indice := 0;
    OK := TRUE;
```

```
    Premier := TRUE;
    IF (CompareString(St,"") = equal) THEN OK := FALSE END;
    IF (CompareString(St,"-") = equal) THEN OK := FALSE END;
    WHILE ((Indice < StringLength (St)) AND OK) DO
      IF ((St[Indice] < "0") OR (St[Indice] > "9"))
         THEN IF (((St[Indice] = ",") OR (St[Indice] = ".")) AND Premier)
                 THEN St[Indice] := ".";
                      Indice := Indice + 1;
                      Premier := FALSE;
                 ELSE IF ((St[Indice] = "-")AND(Indice = StringLength (St)-1))
                        THEN St[Indice] := " ";Indice := Indice + 1;
                        ELSE  OK := FALSE END;
              END;
         ELSE Indice := Indice + 1 END;
    END;
  END Reel;


PROCEDURE CorrPMCL (VAR Phrase : ARRAY OF CHAR;
                          VAR PhrasePron : ARRAY OF CHAR );


BEGIN
    IF TableauChaine.Indice <= 499
    THEN TableauChaine.Tab[TableauChaine.Indice] :=25;

                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;

        TableauChaine.Indice := TableauChaine.Indice + 1;
    END;
    Deb := TRUE;
    CopyString (PPron, PhrasePron);
    IF OpenSpeech () THEN
    QR := FALSE;
    PREM := TRUE;
    ValeurMax := Max;
    ES := "";
    WITH Wp DO
      procGadgetUp := GadgetHandlerSSN;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 0FCAH;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
```

```
cmap[13] := 070FH;
cmap[14] := 0C0EH;
cmap[15] := 0C08H;
cmap[16] := 0620H;
cmap[17] := 0E52H;
cmap[18] := 0A52H;
cmap[19] := 0FCAH;
cmap[20] := 0333H;
cmap[21] := 0444H;
cmap[22] := 0555H;
cmap[23] := 0666H;
cmap[24] := 0777H;
cmap[25] := 0888H;
cmap[26] := 0999H;
cmap[27] := 0AAAH;
cmap[28] := 0CCCH;
cmap[29] := 0DDDH;
cmap[30] := 0EEEH;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

ImgPtr := FindImageTable(66666666H, ImgCount);
IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
   Indice := 0;
   Indlist:= 3;
   WHILE (Indlist <= ImgCount-1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr^[Indlist].Width;
        Height      := ImgPtr^[Indlist].Height;
        Depth       := ImgPtr^[Indlist].Depth;
        ImageData   := ImgPtr^[Indlist].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice:= Indice + 1;
      Indlist := Indlist + 1;
   END;
END;
ImgPtr2 := FindImageTable(11111114H, ImgCount2);
IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
   Indlist:= 6;
   WHILE (Indlist <= ImgCount2-1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr2^[Indlist].Width;
        Height      := ImgPtr2^[Indlist].Height;
        Depth       := ImgPtr2^[Indlist].Depth;
        ImageData   := ImgPtr2^[Indlist].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice:= Indice + 1;
      Indlist := Indlist + 1;
   END;
END;

Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,NIL,Scr,NIL);
Won := CreateWindow (45,50,274,60,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Won # NIL) AND (Win # NIL) THEN
 Rp := Win^.RPort;
 DrawImage(Win^.RPort^,TabImages[10],0,0);
```

```
SetAPen (Rp^,5);
Move (Rp^,0,0);
Draw (Rp^,0,255);
Draw (Rp^,319,255);
Draw (Rp^,319,0);
Draw (Rp^,0,0);
SetAPen (Rp^,1);
Efface := TRUE;
WHILE Efface = TRUE DO
  SA := Sactuel;
  SAP := 0.;
  ValeurPrec := 0.;
  ActivateWindow (Won^);
  BeginGadgetList();
  GadgetTypeReq := TRUE;
  GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                   RelVerify});
  AddGadgetTextButton (147,22,ADR (" OK "));
  AddGadgetTextButton (10,22,ADR ("RECOMMENCER"));
  G1 := EndGadgetList();
  IF (G1 # NIL) THEN
  InitRequester (Req);
  WITH Req DO
    OlderRequest := NIL;
    LeftEdge := 55;
    TopEdge := 120;
    Width := 210;
    Height := 55;
    ReqGadget := G1;
    ReqBorder := NIL;
    ReqText := NIL;
    Flags := RequesterFlagsSet {};
    BackFill := BYTE(10);
    ReqLayer := NIL;
    ImageBMap := NIL;
  END;

     IF CreateConsole (Won^) THEN
      OKNombre := FALSE;
      WHILE NOT OKNombre DO
       wClrScr(Won^);
       wMove (Won^,1,1);PutStr(Won^,ADR(Phrase));
       IF (PREM=TRUE) THEN
         Val := Sinit;
         OKNombre := TRUE;
         wMove (Won^,1,4);
         ConvRealToString (Val,StVal,2,Decimal);
         PutStr (Won^, ADR(StVal));
       ELSE
       wMove (Won^,1,4);
       IF (TableauParametres[1] = 2) THEN
         Res := SayAndReturn (ADR(PhrasePron));
       END;
       wSetCursor (Won^,TRUE);
       GetStr (Won^,ADR (Nombre), SIZE (Nombre));
       wSetCursor (Won^,FALSE);
       Reel (Nombre, OKNombre);
       IF OKNombre THEN Val:= ConvStringToReal (Nombre);
                   END;
       END;
     END;
       PREM := FALSE;
       SAP := SA;
      IF Request (Req, Win^) THEN
      Done := FALSE;
      WHILE (NOT Done) DO
```

```
                Sig := Wait(SignalSet{CARDINAL(Win`.UserPort`.mpSigBit)});
                LOOP
                  Msg := GetMsg(Win^.UserPort^);
                  IF (Msg = NIL) THEN EXIT; END;
                  ProcIMsg (Wp, Msg);
                END;
              END;
            ELSE WriteString ("requester pas créé");END;
            DeleteConsole (Won^);
            END;
            FreeGadgetList (Gl^);
          END;
        END;

        CloseWindow(Win^);
        CloseWindow(Won^);
      END;

   CloseScreen(Scr^);
   END;
Sinit := Val;
CloseSpeech ();
END;
END CorrPMCL;

(*  VAR
   p: ARRAY [0..40] OF CHAR;
   pp: ARRAY [0..40] OF CHAR;

BEGIN
   p := "Combien avez-vous d'argent ?";
   pp := "Cobya ah vay voo darjo ?";
   CorrPMCL (p,pp);    *)

END CorrectionPMCL.
```

```
IMPLEMENTATION MODULE SSNombre;


FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM AmigaDOSProcess IMPORT Delay;
FROM MathLib0 IMPORT entier,real;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, CloseWindow, CloseScreen, ActivateWindow,Image,DrawImage,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, GadgetTypeReq, GlobalGadgetOpt,GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM InOut IMPORT WriteString;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM Strings IMPORT  CopyString, Relation,ConcatString,StringLength,
       CompareString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100, TabNombre,
       TableauTraces,TRecettes,TDepenses,TabNombreInit,TableauParametres,
       Sinit,Sactuel,Max,TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                            ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};


VAR
  Nombre : Str20;
  Done,OKNombre,Efface,Res,QR : BOOLEAN;
  Scr : ScreenPtr;
  Win,Win2,Won,Wun,Wun2 : WindowPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  Gl : GadgetPtr;
  Wp : WindowProc;
  Req : Requester;
  Sig : SignalSet;
  Msg : IntuiMessagePtr;
  ES,Maxaf : Str20;
  PPron : Str100;
  ValeurMax,ValeurPrec,Val,SA,SAP : REAL;
  TabImages : ARRAY [0..59] OF Image;
  ImageRec,ImageDep : Image;
  ImgPtr,ImgPtr2 : ImageDescTablePtr;
  ImgCount,ImgCount2,Indice,Indlist : CARDINAL;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree : REAL;
  DSR: DateStampRecord;
  Rp : RastPortPtr;

PROCEDURE GadgetHandlerSSN (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget
```

```
VAR OK : BOOLEAN;
    V : REAL;
BEGIN

  IF Gad.GadgetID = 0 THEN Done := TRUE; Efface := FALSE;
              IF TableauTraces.Indice <= 499 THEN
                 CopyString (TableauTraces.Tab [TableauTraces.Indice],
                             "Il appuie sur OK");
                 TableauTraces.Indice := TableauTraces.Indice + 1;
              END;
         IF TableauChaine.Indice <= 499
         THEN IF NOT QR
              THEN TableauChaine.Tab [TableauChaine.Indice] := 6;
                   DateStamp(DSR);
                   ActMinute := SHORT (DSR.dsMinute);
                   ActTick := SHORT (DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
                   ree := real(ent);
                   TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                   AncMinute := ActMinute;
                   AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
                   TableauChaine.Tab [TableauChaine.Indice] := 7;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
                   ree := real(ent);
                   TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                   AncMinute := ActMinute;
                   AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
              ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                        = equal)
                   THEN TableauChaine.Tab [TableauChaine.Indice] := 23;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
                   ree := real(ent);
                   TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                   AncMinute := ActMinute;
                   AncTick := ActTick2;
                        TableauChaine.Indice := TableauChaine.Indice + 1;
                        TableauChaine.Tab [TableauChaine.Indice] := 7;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
```

```
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    ELSE TableauChaine.Tab [TableauChaine.Indice] := 15;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                         TableauChaine.Tab [TableauChaine.Indice] := 7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    END;
               END;
          END;

ELSIF Gad.GadgetID = 1
                     THEN  Done := TRUE;
                          IF TableauTraces.Indice <= 499 THEN
                          CopyString (TableauTraces.Tab [TableauTraces.Indice
                                   "Il appuie sur RECOMMENCER");
                          TableauTraces.Indice := TableauTraces.Indice + 1;
                          END;
          IF TableauChaine.Indice <= 499
          THEN IF NOT QR
               THEN TableauChaine.Tab [TableauChaine.Indice] := 5;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
                    TableauChaine.Tab [TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
```

```
        THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
        END;
        ent := ActTick - AncTick;
        ree := real(ent);
        TableauTemps [TableauChaine.Indice] := entier(ree/50.);
        AncMinute := ActMinute;
        AncTick := ActTick2;
        TableauChaine.Indice := TableauChaine.Indice + 1;
    ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
            = equal)
        THEN TableauChaine.Tab [TableauChaine.Indice] := 21;
        DateStamp(DSR);
        ActMinute := SHORT(DSR.dsMinute);
        ActTick := SHORT(DSR.dsTick);
        ActTick2 := ActTick;
        IF ActMinute > AncMinute
        THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
        END;
        ent := ActTick - AncTick;
        ree := real(ent);
        TableauTemps [TableauChaine.Indice] := entier(ree/50.);
        AncMinute := ActMinute;
        AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
             TableauChaine.Tab [TableauChaine.Indice] := 20;
        DateStamp(DSR);
        ActMinute := SHORT(DSR.dsMinute);
        ActTick := SHORT(DSR.dsTick);
        ActTick2 := ActTick;
        IF ActMinute > AncMinute
        THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
        END;
        ent := ActTick - AncTick;
        ree := real(ent);
        TableauTemps [TableauChaine.Indice] := entier(ree/50.);
        AncMinute := ActMinute;
        AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
        ELSE TableauChaine.Tab [TableauChaine.Indice] := 13;
        DateStamp(DSR);
        ActMinute := SHORT(DSR.dsMinute);
        ActTick := SHORT(DSR.dsTick);
        ActTick2 := ActTick;
        IF ActMinute > AncMinute
        THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
        END;
        ent := ActTick - AncTick;
        ree := real(ent);
        TableauTemps [TableauChaine.Indice] := entier(ree/50.);
        AncMinute := ActMinute;
        AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
             TableauChaine.Tab [TableauChaine.Indice] := 12;
        DateStamp(DSR);
        ActMinute := SHORT(DSR.dsMinute);
        ActTick := SHORT(DSR.dsTick);
        ActTick2 := ActTick;
        IF ActMinute > AncMinute
        THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
        END;
        ent := ActTick - AncTick;
        ree := real(ent);
        TableauTemps [TableauChaine.Indice] := entier(ree/50.);
        AncMinute := ActMinute;
        AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
```

```
                END;
            END;
        END;
    ELSE Done := TRUE;
         ES := "quitter";
         Efface := FALSE;
         IF TableauTraces.Indice <= 499 THEN
            CopyString (TableauTraces.Tab [TableauTraces.Indice],
                        "Il appuie sur QUITTER");
            TableauTraces.Indice := TableauTraces.Indice + 1;
         END;
         IF TableauChaine.Indice <= 499
         THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                        = equal)
              THEN TableauChaine.Tab [TableauChaine.Indice] := 22;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
                   ree := real(ent);
                   TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                   AncMinute := ActMinute;
                   AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
                    TableauChaine.Tab [TableauChaine.Indice] := 7;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);

                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
                   ree := real(ent);
                   TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                   AncMinute := ActMinute;
                   AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
              ELSE TableauChaine.Tab [TableauChaine.Indice] := 14;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
                   ree := real(ent);
                   TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                   AncMinute := ActMinute;
                   AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
                    TableauChaine.Tab [TableauChaine.Indice] := 7;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
```

```
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
            END;
  END;
END GadgetHandlerSSN;


PROCEDURE Reel (VAR St : ARRAY OF CHAR ;  VAR OK : BOOLEAN);
  VAR Indice : CARDINAL;
      Premier : BOOLEAN;
  BEGIN
    Indice := 0;
    OK := TRUE;
    Premier := TRUE;
    IF (CompareString(St,"") = equal) THEN OK := FALSE END;
    IF (CompareString(St,"-") = equal) THEN OK := FALSE END;
    WHILE ((Indice < StringLength (St)) AND OK) DO
      IF ((St[Indice] < "0") OR (St[Indice] > "9"))
         THEN IF (((St[Indice] = ",") OR (St[Indice] = ".")) AND Premier)
                 THEN St[Indice] := ".";
                      Indice := Indice + 1;
                      Premier := FALSE;
                 ELSE IF ((St[Indice] = "-")AND(Indice = StringLength (St)-1))
                        THEN St[Indice] := " ";Indice := Indice + 1;
                        ELSE  OK := FALSE END;
              END;
         ELSE Indice := Indice + 1 END;
    END;
  END Reel;


PROCEDURE SaisieSommeNbre(VAR Phrase : ARRAY OF CHAR;
                          VAR PhrasePron : ARRAY OF CHAR;
                          Quit : BOOLEAN; VAR Valeur : REAL;
                          VAR EtatSortie : Str20);

BEGIN
    CopyString (PPron, PhrasePron);
    IF OpenSpeech () THEN
    QR := Quit;
    ValeurMax := Max;
    ES := "";
    WITH Wp DO
      procGadgetUp := GadgetHandlerSSN;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 0FCAH;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
```

```
cmap[15] := 0C08H;
cmap[16] := 0620H;
cmap[17] := 0E52H;
cmap[18] := 0A52H;
cmap[19] := 0FCAH;
cmap[20] := 0333H;
cmap[21] := 0444H;
cmap[22] := 0555H;
cmap[23] := 0666H;
cmap[24] := 0777H;
cmap[25] := 0888H;
cmap[26] := 0999H;
cmap[27] := 0AAAH;
cmap[28] := 0CCCH;
cmap[29] := 0DDDH;
cmap[30] := 0EEEH;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

ImgPtr := FindImageTable(66666666H, ImgCount);
IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
   Indice := 0;
   Indlist := 3;
   WHILE (Indlist <= ImgCount - 1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr^[Indlist].Width;
        Height      := ImgPtr^[Indlist].Height;
        Depth       := ImgPtr^[Indlist].Depth;
        ImageData   := ImgPtr^[Indlist].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice := Indice + 1;
      Indlist := Indlist + 1;
   END;
END;
ImgPtr2 := FindImageTable(11111114H, ImgCount2);
IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
   Indlist := 6;
   WHILE (Indlist <= ImgCount2 - 1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr2^[Indlist].Width;
        Height      := ImgPtr2^[Indlist].Height;
        Depth       := ImgPtr2^[Indlist].Depth;
        ImageData   := ImgPtr2^[Indlist].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice := Indice + 1;
      Indlist := Indlist + 1;
   END;
END;
Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,NIL,Scr,NIL);
Won := CreateWindow (45,50,274,60,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Won # NIL) AND (Win # NIL) THEN
 Rp := Win^.RPort;
 IF NOT QR THEN
    DrawImage(Win^.RPort^,TabImages[10],0,0);
    SetAPen (Rp^,5);
 END;
```

```
IF QR THEN
   IF (CompareString (PPron,"Cobya ah vay voo daypensay ?") = equal)
   THEN
      DrawImage(Win^.RPort^,TabImages[5],0,0);
      SetAPen (Rp^,2);
   ELSE
      DrawImage(Win^.RPort^,TabImages[4],0,0);
      SetAPen (Rp^,8);
   END;
END;
Move (Rp^,0,0);
Draw (Rp^,0,255);
Draw (Rp^,319,255);
Draw (Rp^,319,0);
Draw (Rp^,0,0);
SetAPen (Rp^,1);
Efface := TRUE;
WHILE Efface = TRUE DO
  SA := Sactuel;
  SAP := 0.;
  ValeurPrec := 0.;
  ActivateWindow (Won^);
  BeginGadgetList();
  GadgetTypeReq := TRUE;
  GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                   RelVerify});
  AddGadgetTextButton (137,22,ADR (" OK "));
  AddGadgetTextButton (5,22,ADR ("RECOMMENCER"));
  IF Quit = TRUE THEN AddGadgetTextButton (215,22,ADR ("QUITTER")) END;
  G1 := EndGadgetList();
  IF (G1 # NIL) THEN
  InitRequester (Req);
  WITH Req DO
    OlderRequest := NIL;
    LeftEdge := 10;
    TopEdge := 120;
    Width := 300;
    Height := 55;
    ReqGadget := G1;
    ReqBorder := NIL;
    ReqText := NIL;
    Flags := RequesterFlagsSet {};
    BackFill := BYTE(10);
    ReqLayer := NIL;
    ImageBMap := NIL;
  END;

     IF CreateConsole (Won^) THEN
      OKNombre := FALSE;
      WHILE NOT OKNombre DO
       wClrScr(Won^);
       wMove (Won^,1,1);PutStr(Won^,ADR(Phrase));
       wMove (Won^,1,4);
       IF (TableauParametres[1] = 2) THEN
         Res := SayAndReturn (ADR(PhrasePron));
       END;
       wSetCursor (Won^,TRUE);
       GetStr (Won^,ADR (Nombre), SIZE (Nombre));
       wSetCursor (Won^,FALSE);
       Reel (Nombre, OKNombre);
       IF OKNombre THEN Val:= ConvStringToReal (Nombre);
                   END;
       END;
       SAP := SA;
      IF Request (Req, Win^) THEN
      Done := FALSE;
```

```
            WHILE (NOT Done) DO
              Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
              LOOP
                Msg := GetMsg(Win^.UserPort^);
                IF (Msg = NIL) THEN EXIT; END;
                ProcIMsg (Wp, Msg);
              END;
            END;
           ELSE WriteString ("requester pas créé");END;
           DeleteConsole (Won^);
           END;
           FreeGadgetList (Gl^);
         END;
       END;
       EtatSortie := ES;
       CloseWindow(Win^);
       CloseWindow(Won^);
     END;

   CloseScreen(Scr^);
   END;
Valeur := Val;
CloseSpeech ();
END;
END SaisieSommeNbre;

(*  VAR
    p : ARRAY [0..40] OF CHAR;
    pp : ARRAY [0..40] OF CHAR;
    q : BOOLEAN;
    v : REAL;
    e : Str20;

BEGIN
    p := "Combien avez-vous dépensé ?";
    pp := "Cobya ah vay voo daypensay ?";
    q := TRUE;
    SaisieSommeNbre (p,pp,q,v,e);                *)

END SSNombre.
```

```
IMPLEMENTATION MODULE SSBillets;

FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton,AddGadgetImageButton, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40, TabNombre,TableauTraces,
        TRecettes,TDepenses,TabNombreInit,Max,Sactuel,Sinit,Str100,
        TableauParametres,TempsSIBillets,TableauChaine,AncMinute,AncTick,
        TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                    ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Utilitaires IMPORT ConvPrononcable;
FROM Drawing IMPORT RectFill,SetAPen,Draw,Move;
FROM Text IMPORT Text;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre,DoneRec,Recom : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff,QR : BOOLEAN;
  For : RealToStringFormat;
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Recette,Depense,EtatPM,FIN : Image;
  Win2,Win,Won,Wun : WindowPtr;
```

```
  Mini : ARRAY [0..8] OF Image;
  Monnaie : ARRAY [0..8] OF Image;
  Commande: ARRAY [0..5] OF Image;
  Icone: ARRAY [0..3] OF Image;
  ImgPtr,ImgPtr6,ImgPtr7,ImgPtr8 : ImageDescTablePtr;
  ImgCount,ImgCount6,ImgCount7,ImgCount8,Indice,list6,list7, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  G1,G1Eff,G1Qui,G1Fin,G1OK,G1Rec : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK,WpRec : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK,ReqRec : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK,MsgRec : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50,Nb20,Nb5,Nb1 : CARDINAL;
  B100, B50, B20, P5, P1, IRecettes,IDepenses,Fait,I : INTEGER;
  Somme, SommePrecedente, SommeMax ,SA,SAP: REAL;
  StrSomme,ES,fonc,Maxaf : Str20;
  Res,Quit : BOOLEAN;
  DerPron,PPron : Str100;
  DSR : DateStampRecord;
  IndTemps, II : CARDINAL;
  Nbillets : INTEGER;
  NminBEG, NticksBEG,NticksEND,NminEND,TempsInterm : ARRAY [1..20] OF INTEGER;
  ent : INTEGER;
  re2,ree, ree2 : REAL;
  ActMinute,ActTick,ActTick2 : INTEGER;
  Rp :RastPortPtr;

PROCEDURE GadgetHandlerEffBB (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Efface := TRUE ; DoneEff := TRUE;
           IF TableauChaine.Indice <= 499
           THEN IF NOT QR
                THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                            = equal)
                     THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                          TableauChaine.Indice := TableauChaine.Indice + 1;
                     ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                    DateStamp(DSR);
```

```
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                 END;
           END;
       END;
      Trace := "Il confirme la demande ";
      IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END; |
1 : Efface := FALSE; DoneEff := TRUE;
      IF TableauChaine.Indice <= 499
      THEN IF NOT QR
           THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                        = equal)
                THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
```

```
                              TableauChaine.Indice := TableauChaine.Indice + 1;
                         END;
                    END;
               END;
               Trace := "Il infirme la demande ";
               IF TableauTraces.Indice <= 499 THEN
                    CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                    TableauTraces.Indice :=  TableauTraces.Indice + 1;
               END; |
     2 :
  END;
END GadgetHandlerEffBB;

PROCEDURE GadgetHandlerQuiBB (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Quitter := TRUE ; DoneQui := TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab[TableauChaine.Indice] := 7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
          Trace := "Il confirme la demande ";
          IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
          END; |
     1 : Quitter := FALSE; DoneQui := TRUE;
          IF TableauChaine.Indice <= 499
          THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                         = equal)
               THEN TableauChaine.Tab [TableauChaine.Indice] := 20;

                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;                         TableauChaine.In
ce := TableauChaine.Indice + 1;
               ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
```

```
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
              END;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
    2 :
  END;
END GadgetHandlerQuiBB;

PROCEDURE GadgetHandlerFinBB (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
    0 : Fini := TRUE ; DoneFin := TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab[TableauChaine.Indice] := 7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
          Trace := "Il confirme la demande ";
          IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
          END |

    1 : Fini := FALSE; DoneFin := TRUE;
          IF TableauChaine.Indice <= 499
          THEN IF NOT QR
               THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
               ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                              = equal)
                     THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
```

```
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                          TableauChaine.Indice := TableauChaine.Indice + 1;
                     ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                          TableauChaine.Indice := TableauChaine.Indice + 1;
                     END;
               END;
          END;
          Trace := "Il infirme la demande ";
          IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
          END; |
    2 :
  END;
END GadgetHandlerFinBB;

PROCEDURE GadgetHandlerRecBB (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget)
BEGIN
  CASE Gad.GadgetID OF
     0 : Recom := TRUE ; DoneRec := TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
          Trace := "Il confirme la demande ";
          IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
          END; |
    1 : Recom := FALSE; DoneRec := TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
```

```
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
      2 :
   END;
END GadgetHandlerRecBB;

PROCEDURE GadgetHandlerBB (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
    SomAct : REAL;
BEGIN
   IF OpenSpeech () THEN
   CASE Gad.GadgetID OF
      0 : DateStamp(DSR);
          NminEND[IndTemps]:=SHORT (DSR.dsMinute);
          NticksEND[IndTemps] := SHORT (DSR.dsTick);
          Nbillets := Nbillets + 1;
          ActivEff := TRUE;
          Nb5000 := Nb5000 + 1;
          SommePrecedente := Somme;
          Somme := Somme + 5000.;
          IF Nb5000 <= 7
          THEN
             ree := real (Nb5000);
             ree2 := ree/2.;
             ent := entier (ree2);
             re2 := real (ent);
             IF ree2 = re2
             THEN
               Der.Y := 26;
               Der.X := 189 + ((Nb5000 - 2) * 18);
               Der.X := Der.X + ((Nb5000 - 2) DIV 2);
             ELSE
               Der.Y := 33;
               Der.X := 170 + ((Nb5000 - 1) * 19);
               Der.X := Der.X - ((Nb5000 - 1) DIV 2);
             END;
             Der.Type := "BIL";
             Der.Last := 5000.;
             AuMoinsUn := TRUE;
             Delay (25);
             DrawImage (Win^.RPort^,Mini[0],Der.X,Der.Y)
          ELSE
             AuMoinsUn := FALSE;
             Der.Last := 5000.
          END;
          IF (TableauParametres[1] = 2) THEN
             Res := SayAndReturn (ADR("senk myl froh"));
             DerPron := "senk myl froh";
          END;
          wMove (Won^,7,1);
          wClrEndLine (Won^);
```

```
      Somme := Somme + 500.;
      IF Nb500 <= 8
      THEN
         ree := real (Nb500);
         ree2 := ree/2.;
         ent := entier (ree2);
         re2 := real (ent);
         IF ree2 = re2
         THEN
           Der.Y := 84;
           Der.X := 187 + ((Nb500 - 2) * 16);
           Der.X := Der.X + ((Nb500 - 2) DIV 2);
         ELSE
           Der.Y := 91;
           Der.X := 170 + ((Nb500 - 1) * 17);
           Der.X := Der.X - ((Nb500 - 1) DIV 2);
         END;
         Der.Type := "BIL";
         Der.Last := 500.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,Mini[2],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 500.
      END;
      IF (TableauParametres[1] = 2) THEN
         Res := SayAndReturn (ADR("senk soh froh"));
         DerPron := "senk san froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 0, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 500 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
3 :   DateStamp(DSR);
      NminEND[IndTemps]:=SHORT (DSR.dsMinute);
      NticksEND[IndTemps] := SHORT (DSR.dsTick);
      Nbillets := Nbillets + 1;
      ActivEff := TRUE;
      Nb100 := Nb100 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 100.;
      IF Nb100 <= 8
      THEN
         ree := real (Nb100);
         ree2 := ree/2.;
         ent := entier (ree2);
         re2 := real (ent);
         IF ree2 = re2
         THEN
           Der.Y := 113;
           Der.X := 186 + ((Nb100 - 2) * 15);
           Der.X := Der.X + ((Nb100 - 2) DIV 2);
         ELSE
           Der.Y := 120;
           Der.X := 170 + ((Nb100 - 1) * 16);
           Der.X := Der.X - ((Nb100 - 1) DIV 2);
         END;
```

```
      Der.Type := "BIL";
      Der.Last := 100.;
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,Mini[3],Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 100.
   END;
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("soh froh"));
      DerPron := "soh froh";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 0, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);
   Trace := "Il appuie sur 100 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END |
4 : DateStamp(DSR);
   NminEND[IndTemps]:=SHORT (DSR.dsMinute);
   NticksEND[IndTemps] := SHORT (DSR.dsTick);
   Nbillets := Nbillets + 1;
   ActivEff := TRUE;
   Nb50 := Nb50 + 1;
   SommePrecedente := Somme;
   Somme := Somme + 50.;
   IF Nb50 <= 8
   THEN
      Der.X := 170 + ((Nb50 - 1) * 18);
      Der.Y := 145;
      Der.Type := "PA";
      Der.Last := 50.;
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,Mini[4],Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 50.
   END;
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("sekan't froh"));
      DerPron := "sekan't froh";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 0, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);
   Trace := "Il appuie sur 50 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END |
5 : DateStamp(DSR);
   NminEND[IndTemps]:=SHORT (DSR.dsMinute);
   NticksEND[IndTemps] := SHORT (DSR.dsTick);
   Nbillets := Nbillets + 1;
```

```
      ActivEff := TRUE;
      Nb20 := Nb20 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 20.;
      IF Nb20 <= 8
      THEN
         Der.X := 170 + ((Nb20 - 1) * 18);
         Der.Y := 168;
         Der.Type := "PA";
         Der.Last := 20.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,Mini[5],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 20.
      END;
      IF (TableauParametres[1] = 2) THEN
         Res := SayAndReturn (ADR("vant froh"));
         DerPron := "vant froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 0, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 20 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  6 : DateStamp(DSR);
      NminEND[IndTemps]:=SHORT (DSR.dsMinute);
      NticksEND[IndTemps] := SHORT (DSR.dsTick);
      Nbillets := Nbillets + 1;
      ActivEff := TRUE;
      Nb5 := Nb5 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 5.;
      IF Nb5 <= 8
      THEN
         Der.X := 170 + ((Nb5 - 1) * 18);
         Der.Y := 194;
         Der.Type := "PA";
         Der.Last := 5.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,Mini[6],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 5.
      END;
      IF (TableauParametres[1] = 2) THEN
         Res := SayAndReturn (ADR("senk froh"));
         DerPron := "senk froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 0, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 5 Frs ";
```

```
        IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ;
    7 : DateStamp(DSR);
        NminEND[IndTemps]:=SHORT (DSR.dsMinute);
        NticksEND[IndTemps] := SHORT (DSR.dsTick);
        Nbillets := Nbillets + 1;
        ActivEff := TRUE;
        Nb1 := Nb1 + 1;
        SommePrecedente := Somme;
        Somme := Somme + 1.;
        IF Nb1 <= 8
        THEN
           Der.X := 170 + ((Nb1 - 1) * 18);
           Der.Y := 218;
           Der.Type := "PA";
           Der.Last := 1.;
           AuMoinsUn := TRUE;
           Delay (25);
           DrawImage (Win^.RPort^,Mini[7],Der.X,Der.Y)
        ELSE
           AuMoinsUn := FALSE;
           Der.Last := 1.
        END;
        IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("an froh"));
           DerPron := "an froh";
        END;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 0, For);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        Trace := "Il appuie sur 1 Frs ";
        IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ;
    8 : IF ActivEff THEN
         IF (NminEND[IndTemps] > NminBEG[IndTemps]) THEN
          NticksEND[IndTemps] := NticksEND[IndTemps] + (3000 * (NminEND[IndTe
s] - NminBEG[IndTemps]));
         END;
         ent := (NticksEND[IndTemps] - NticksBEG[IndTemps]);
         ree := real (ent);
         ree2 := (ree / 50.);
         TempsInterm[IndTemps] := entier (ree2);
         IF IndTemps < 20 THEN
           IndTemps := IndTemps + 1;
         END;
          IF TableauChaine.Indice <= 499
          THEN IF NOT QR
               THEN TableauChaine.Tab[TableauChaine.Indice] := 4;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
                   ree := real(ent);
                   TableauTemps [TableauChaine.Indice] := entier(ree/50.);
```

```
            AncMinute := ActMinute;
            AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
        ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                    = equal)
             THEN TableauChaine.Tab [TableauChaine.Indice] := 21;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
                  TableauChaine.Indice := TableauChaine.Indice + 1;
             ELSE TableauChaine.Tab [TableauChaine.Indice] := 13;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
                  TableauChaine.Indice := TableauChaine.Indice + 1;
             END;
        END;
    END;
Trace := "Il demande d'effacer ";
IF TableauTraces.Indice <= 499 THEN
     CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
     TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                  RelVerify});
AddGadgetTextButton (10,34, ADR("OUI"));
AddGadgetTextButton (140,34, ADR("NON"));
AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS EFFACER ?"));
GadgetOpt (GadgetFlagsSet{}, GadgetActivationSet {RelVerify},
           GadgetMutualExcludeSet {});
GlEff := EndGadgetList ();
InitRequester (ReqEff);
WITH ReqEff DO
       OlderRequest := NIL;
       LeftEdge := 70;
       TopEdge := 22;
       Width := 180;
       Height := 50;
       ReqGadget := GlEff;
       ReqBorder := NIL;
       ReqText := NIL;
       Flags := RequesterFlagsSet {};
       BackFill := BYTE(10);
       ReqLayer := NIL;
       ImageBMap := NIL;
END;
```

```
IF Request (ReqEff. Win^) THEN
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("vooley voo effassay ?"));
   END;
   DoneEff := FALSE;
   WHILE NOT DoneEff DO
       Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
       LOOP
          MsgEff := GetMsg (Win^.UserPort^);
          IF (MsgEff = NIL) THEN EXIT; END;
          ProcIMsg (WpEff, MsgEff);
       END;
   END;
END;
FreeGadgetList (GlEff^);
Delay(25);
IF Efface THEN
          IF Der.Last = 5000.
          THEN
            Nb5000 := Nb5000 - 1;
          ELSIF Der.Last = 1000.
          THEN
           Nb1000 := Nb1000 - 1;
          ELSIF Der.Last = 500.
          THEN
            Nb500 := Nb500 - 1;
          ELSIF Der.Last = 100.
          THEN
            Nb100 := Nb100 - 1;
          ELSIF Der.Last = 50.
          THEN
            Nb50 := Nb50 - 1;
          ELSIF Der.Last = 20.
          THEN
            Nb20 := Nb20 - 1;
          ELSIF Der.Last = 5.
          THEN
            Nb5 := Nb5 - 1;
          ELSIF Der.Last = 1.
          THEN
            Nb1 := Nb1 - 1;
          END;

  ActivEff := FALSE;
  IF ((CompareString (Der.Type,"BIL") = equal) AND (AuMoinsUn))
  THEN
      SetAPen (Rp^,0);
      IF Der.Last = 500.
      THEN
        RectFill (Rp^,Der.X,Der.Y,Der.X+31,Der.Y+13);
      ELSE
        RectFill (Rp^,Der.X,Der.Y,Der.X+35,Der.Y+13);
      END;
      SetAPen (Rp^,1);
      IF (Der.X # 170) THEN
        IF Der.Last = 5000.
          THEN
            ree := real (Nb5000);
            ree2 := ree/2.;
            ent := entier (ree2);
            re2 := real (ent);
            IF ree2 = re2
            THEN
            Der.Y := 26;
            Der.X := 189 + ((Nb5000 - 2) * 18);
            Der.X := Der.X + ((Nb5000 - 2) DIV 2);
```

```
                ELSE
                Der.Y := 33;
                Der.X := 170 + ((Nb5000 - 1) * 19);
                Der.X := Der.X - ((Nb5000 - 1) DIV 2);
                END;
                DrawImage (Win^.RPort^,Mini[0],Der.X,Der.Y)
              ELSIF Der.Last = 1000.
              THEN
               ree := real (Nb1000);
               ree2 := ree/2.;
               ent := entier (ree2);
               re2 := real (ent);
               IF ree2 = re2
               THEN
                 Der.Y := 55;
                 Der.X := 188 + ((Nb1000 - 2) * 17);
                 Der.X := Der.X + ((Nb1000 - 2) DIV 2);
               ELSE
                 Der.Y := 62;
                 Der.X := 170 + ((Nb1000 - 1) * 18);
                 Der.X := Der.X - ((Nb1000 - 1) DIV 2);
               END;
               DrawImage (Win^.RPort^,Mini[1],Der.X,Der.Y)
              ELSIF Der.Last = 500.
              THEN
                ree := real (Nb500);
                ree2 := ree/2.;
                ent := entier (ree2);
                re2 := real (ent);
                IF ree2 = re2
                THEN
                  Der.Y := 84;
                  Der.X := 187 + ((Nb500 - 2) * 16);
                  Der.X := Der.X + ((Nb500 - 2) DIV 2);
                ELSE
                  Der.Y := 91;
                  Der.X := 170 + ((Nb500 - 1) * 17);
                  Der.X := Der.X - ((Nb500 - 1) DIV 2);
                END;
                DrawImage (Win^.RPort^,Mini[2],Der.X,Der.Y)
              ELSIF Der.Last = 100.
              THEN
                ree := real (Nb100);
                ree2 := ree/2.;
                ent := entier (ree2);
                re2 := real (ent);
                IF ree2 = re2
                THEN
                  Der.Y := 113;
                  Der.X := 186 + ((Nb100 - 2) * 15);
                  Der.X := Der.X + ((Nb100 - 2) DIV 2);
                ELSE
                  Der.Y := 120;
                  Der.X := 170 + ((Nb100 - 1) * 16);
                  Der.X := Der.X - ((Nb100 - 1) DIV 2);
                END;
                DrawImage (Win^.RPort^,Mini[3],Der.X,Der.Y)

          END
        END
    ELSIF ((CompareString (Der.Type,"PA") = equal) AND (AuMoinsUn))
    THEN SetAPen (Rp^,0);
         RectFill (Rp^,Der.X,Der.Y,Der.X +16,Der.Y+16);
         SetAPen (Rp^,1);

    END;
```

```
        SommePrecedente := Somme;
        Somme := Somme - Der.Last;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 0, For);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        DerPron := "";
      END;
      DateStamp(DSR);
      NminBEG[IndTemps] := SHORT(DSR.dsMinute);
      NminEND[IndTemps] := NminBEG[IndTemps];
      NticksBEG[IndTemps] := SHORT(DSR.dsTick);
      NticksEND[IndTemps] := NticksBEG[IndTemps];
    END ;

9:  IF (NminEND[IndTemps] > NminBEG[IndTemps]) THEN
      NticksEND[IndTemps] := NticksEND[IndTemps] +
        (3000 * (NminEND[IndTemps] - NminBEG[IndTemps]));
     END;
     ent := (NticksEND[IndTemps] - NticksBEG[IndTemps]);
     ree := real (ent);
     ree2 := (ree / 50.);
     TempsInterm[IndTemps] := entier (ree2);
     IF IndTemps < 20 THEN
       IndTemps := IndTemps + 1;
     END;
      IF TableauChaine.Indice <= 499
      THEN IF NOT QR
           THEN TableauChaine.Tab[TableauChaine.Indice] := 6;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                       = equal)
                THEN TableauChaine.Tab [TableauChaine.Indice] := 23;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE TableauChaine.Tab [TableauChaine.Indice] := 15;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
```

```
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                 END;
          END;
    END;
Trace := "Il appuie sur OK  ";
IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                  RelVerify});
AddGadgetTextButton (10,34, ADR("OUI"));
AddGadgetTextButton (140,34, ADR("NON"));
AddGadgetTextButton (8,9, ADR("EST-CE BIEN JUSTE ?"));
GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
           GadgetMutualExcludeSet {});
G1Fin := EndGadgetList ();
InitRequester (ReqFin);
WITH ReqFin DO
        OlderRequest := NIL;
        LeftEdge := 70;
        TopEdge := 22;
        Width := 180;
        Height := 50;
        ReqGadget := G1Fin;
        ReqBorder := NIL;
        ReqText := NIL;
        Flags := RequesterFlagsSet {};
        BackFill := BYTE(10);
        ReqLayer := NIL;
        ImageBMap := NIL;
END;
IF Request (ReqFin, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("avay voo feeny ?"));
   END;
   DoneFin := FALSE;
   WHILE NOT DoneFin DO
        Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
        LOOP
           MsgFin := GetMsg (Win^.UserPort^);
           IF (MsgFin = NIL) THEN EXIT; END;
           ProcIMsg (WpFin, MsgFin);
        END;
   END;
END;

FreeGadgetList (G1Fin^);
Delay(25);
IF Fini
THEN Done := TRUE
ELSE
  DateStamp(DSR);
  NminBEG[IndTemps] := SHORT(DSR.dsMinute);
  NminEND[IndTemps] := NminBEG[IndTemps];
  NticksBEG[IndTemps] := SHORT(DSR.dsTick);
```

```
        NticksEND[IndTemps] := NticksBEG[IndTemps];

      END;
10 :IF QR = TRUE
    THEN
      IF (NminEND[IndTemps] > NminBEG[IndTemps]) THEN
       NticksEND[IndTemps] := NticksEND[IndTemps] +
                    (3000 * (NminEND[IndTemps] - NminBEG[IndTemps]));
      END;
      ent := (NticksEND[IndTemps] - NticksBEG[IndTemps]);
      ree := real (ent);
      ree2 := (ree / 50.);
      TempsInterm[IndTemps] := entier (ree2);
      IF IndTemps < 20 THEN
        IndTemps := IndTemps + 1;
      END;
       IF TableauChaine.Indice <= 499
       THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                        = equal)
            THEN TableauChaine.Tab [TableauChaine.Indice] := 22;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
            ELSE TableauChaine.Tab [TableauChaine.Indice] := 14;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
       END;
     Trace := "Il appuie sur QUITTER ";
     IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
     END;
     BeginGadgetList ();
     GadgetTypeReq := TRUE;
     GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                      RelVerify});
     AddGadgetTextButton (10,34, ADR("OUI"));
     AddGadgetTextButton (140,34, ADR("NON"));
     AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS QUITTER ?"));
     GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                GadgetMutualExcludeSet {});
     GlQui := EndGadgetList ();
     InitRequester (ReqQui);
     WITH ReqQui DO
```

```
        OlderRequest := NIL;
        LeftEdge := 70;
        TopEdge := 22;
        Width := 180;
        Height := 50;
        ReqGadget := GlQui;
        ReqBorder := NIL;
        ReqText := NIL;
        Flags := RequesterFlagsSet {};
        BackFill := BYTE(10);
        ReqLayer := NIL;
        ImageBMap := NIL;
 END;
 IF Request (ReqQui, Win^) THEN
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("voolay voo kittay ?"));
    END;
    DoneQui := FALSE;
    WHILE NOT DoneQui DO
        Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
        LOOP
           MsgQui := GetMsg (Win^.UserPort^);
           IF (MsgQui = NIL) THEN EXIT; END;
           ProcIMsg (WpQui, MsgQui);
        END;
    END;
 END;
 FreeGadgetList (GlQui^);
 Delay(25);
 IF Quitter
 THEN Done := TRUE;ES := "quitter";
 ELSE
   DateStamp(DSR);
   NminBEG[IndTemps] := SHORT(DSR.dsMinute);
   NminEND[IndTemps] := NminBEG[IndTemps];
   NticksBEG[IndTemps] := SHORT(DSR.dsTick);
   NticksEND[IndTemps] := NticksBEG[IndTemps];
 END
ELSE
  IF (NminEND[IndTemps] > NminBEG[IndTemps]) THEN
   NticksEND[IndTemps] := NticksEND[IndTemps] +
                      (3000 * (NminEND[IndTemps] - NminBEG[IndTemps]));
  END;
  ent := (NticksEND[IndTemps] - NticksBEG[IndTemps]);
  ree := real (ent);
  ree2 := (ree / 50.);
  TempsInterm[IndTemps] := entier (ree2);
  IF IndTemps < 20 THEN
    IndTemps := IndTemps + 1;
  END;
   IF TableauChaine.Indice <= 499
   THEN TableauChaine.Tab[TableauChaine.Indice] := 5;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
       TableauChaine.Indice := TableauChaine.Indice + 1;
   END;
```

```
Trace := 'Il appuie sur RECOMMENCER ';
IF TableauTraces.Indice <= 499 THEN
     CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
     TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                 RelVerify});
AddGadgetTextButton (10,34, ADR("OUI"));
AddGadgetTextButton (180,34, ADR("NON"));
AddGadgetTextButton (4,9, ADR("VOULEZ-VOUS RECOMMENCER ?"));
GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
         GadgetMutualExcludeSet {});
GlRec := EndGadgetList ();
InitRequester (ReqRec);
WITH ReqRec DO
     OlderRequest := NIL;
     LeftEdge := 50;
     TopEdge := 22;
     Width := 220;
     Height := 50;
     ReqGadget := GlRec;
     ReqBorder := NIL;
     ReqText := NIL;
     Flags := RequesterFlagsSet {};
     BackFill := BYTE(10);
     ReqLayer := NIL;
     ImageBMap := NIL;
END;
IF Request (ReqRec, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("Voolay voo ra comohsay ?"));
   END;
   DoneRec := FALSE;
   WHILE NOT DoneRec DO
     Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
     LOOP
        MsgRec := GetMsg (Win^.UserPort^);
        IF (MsgRec = NIL) THEN EXIT; END;
        ProcIMsg (WpRec, MsgRec);
     END;
 END;
END;
FreeGadgetList (GlRec^);
Delay(25);
IF Recom THEN
     SetAPen (Rp^,0);
     RectFill (Rp^,170,22,317,236);
     SetAPen (Rp^,1);
     Somme := 0.;
     wMove (Won^,7,1);
     wClrEndLine (Won^);
     ConvRealToString (Somme, StrSomme, 2, For);
     PutStr (Won^,ADR(StrSomme));
     wMove (Won^,15,1);
     PutStr (Won^,ADR("Frs"));
     wSetCursor (Won^,FALSE);
     ES := "";
     SommeMax := Max;
     Nb100 := 0;
     Nb50 := 0;
     Nb20 := 0;
     Nb1000 := 0;
     Nb500 := 0;
     Nb5000 := 0;
```

```
                Nb5 := 0;
                Nb1 := 0;
                Der.Last := 0.;
                Der.X := 150;
                AuMoinsUn := FALSE;
                ActivEff := FALSE;
                For := Decimal;
                CopyString (DerPron,PPron);
              END;
            DateStamp(DSR);
            NminBEG[IndTemps] := SHORT(DSR.dsMinute);
            NminEND[IndTemps] := NminBEG[IndTemps];
            NticksBEG[IndTemps] := SHORT(DSR.dsTick);
            NticksEND[IndTemps] := NticksBEG[IndTemps];
             END;
      11 : IF (TableauParametres[1] = 2) THEN
                Res := SayAndReturn (ADR(DerPron));
              END;
  END;
CloseSpeech ();
END;
END GadgetHandlerBB;

(*****************************************************************************)
(*                              SaisieSommeBillets                           *)
(*****************************************************************************)

PROCEDURE SaisieSommeBillets (VAR Phrase : ARRAY OF CHAR;
                              VAR PhrasePron : ARRAY OF CHAR;
                              Quit : BOOLEAN; VAR Valeur : REAL;
                              VAR EtatSortie : Str20);


BEGIN
    CopyString (PPron,PhrasePron);
    ES := "";
    QR := Quit;
    Somme := 0.;
    SommeMax := Max;
    Nb100 := 0;
    Nb50 := 0;
    Nb20 := 0;
    Nb1000 := 0;
    Nb500 := 0;
    Nb5000 := 0;
    Nb5 := 0;
    Nb1 := 0;
    Der.Last := 0.;
    Der.X := 150;
    AuMoinsUn := FALSE;
    ActivEff := FALSE;
    For := Decimal;
    Nbillets := 0;
    IndTemps := 1;
    WITH Wp DO
      procGadgetUp := GadgetHandlerBB;
    END;
    WITH WpEff DO
      procGadgetUp := GadgetHandlerEffBB;
    END;
    WITH WpQui DO
      procGadgetUp := GadgetHandlerQuiBB;
    END;
    WITH WpFin DO
      procGadgetUp := GadgetHandlerFinBB;
    END;
```

```
WITH WpRec DO
  procGadgetUp := GadgetHandlerRecBB;
END;
Scr := CreateScreen (320,256,5,NIL);
IF (Scr # NIL) THEN
  cmap[00] := 0000H;
  cmap[01] := 0FFFH;
  cmap[02] := 0E00H;
  cmap[03] := 0A00H;
  cmap[04] := 0D80H;
  cmap[05] := 0FE0H;
  cmap[06] := 0FCAH;
  cmap[07] := 0080H;
  cmap[08] := 00B6H;
  cmap[09] := 00DDH;
  cmap[10] := 00AFH;
  cmap[11] := 007CH;
  cmap[12] := 000FH;
  cmap[13] := 070FH;
  cmap[14] := 0C0EH;
  cmap[15] := 0C08H;
  cmap[16] := 0620H;
  cmap[17] := 0E52H;
  cmap[18] := 0A52H;
  cmap[19] := 0FCAH;
  cmap[20] := 0333H;
  cmap[21] := 0444H;
  cmap[22] := 0555H;
  cmap[23] := 0666H;
  cmap[24] := 07D8H;
  cmap[25] := 0C97H;
  cmap[26] := 09CFH;
  cmap[27] := 0D9EH;
  cmap[28] := 0EB0H;
  cmap[29] := 0DDDH;
  cmap[30] := 0EC7H;
  cmap[31] := 0FFFH;
  LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

  ImgPtr := FindImageTable(15111111H, ImgCount);
  IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
    Indice := 0;
    WHILE (Indice <= ImgCount-1) DO
      WITH Mini[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr^[Indice].Width;
        Height      := ImgPtr^[Indice].Height;
        Depth       := ImgPtr^[Indice].Depth;
        ImageData   := ImgPtr^[Indice].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice := Indice + 1;
    END;

  ImgPtr6 := FindImageTable(12111111H, ImgCount6);
  IF (ImgPtr6 # NIL) AND (ImgCount6 # 0) THEN
    Indice := 0;
    WHILE (Indice <= ImgCount6 - 1) DO
      WITH Monnaie[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr6^[Indice].Width;
        Height      := ImgPtr6^[Indice].Height;
```

```
      Depth       := ImgPtr6^[Indice].Depth;
      ImageData   := ImgPtr6^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

ImgPtr7 := FindImageTable(13111111H, ImgCount7);
IF (ImgPtr7 # NIL) AND (ImgCount7 # 0) THEN
  Indice := 0;
  WHILE (Indice <= ImgCount7 -1) DO
    WITH Commande [Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr7^[Indice].Width;
      Height      := ImgPtr7^[Indice].Height;
      Depth       := ImgPtr7^[Indice].Depth;
      ImageData   := ImgPtr7^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

ImgPtr8 := FindImageTable(14111111H, ImgCount8);
IF (ImgPtr8 # NIL) AND (ImgCount8 # 0) THEN
  Indice := 0;
  WHILE (Indice <= ImgCount8 -1) DO
    WITH Icone[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr8^[Indice].Width;
      Height      := ImgPtr8^[Indice].Height;
      Depth       := ImgPtr8^[Indice].Depth;
      ImageData   := ImgPtr8^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

   BeginGadgetList();
   AddGadgetImageButton (10,59,Monnaie[0]);
   AddGadgetImageButton (12,99,Monnaie[1]);
   AddGadgetImageButton (14,139,Monnaie[2]);
   AddGadgetImageButton (16,179,Monnaie[3]);
   AddGadgetImageButton (93,60,Monnaie[4]);
   AddGadgetImageButton (90,97,Monnaie[5]);
   AddGadgetImageButton (92,139,Monnaie[6]);
   AddGadgetImageButton (96,182,Monnaie[7]);
   AddGadgetImageButton (3,223,Commande[0]);

   IF Quit = TRUE
   THEN
       AddGadgetImageButton (63,223,Commande[2]);
       AddGadgetImageButton (102,219,Commande[3]);
   ELSE
       AddGadgetImageButton (125,223,Commande[2]);
       AddGadgetImageButton (65,223,Commande[1])
   END;
   IF TableauParametres[1] = 2 THEN
           AddGadgetImageButton (136,175,Commande[4])
```

```
END;
Gl := EndGadgetList();
IF (Gl # NIL) THEN
   Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,Gl,Scr,NIL);
   Won := CreateWindow (167,239,151,15,WIDCMP,WFlags2,NIL,Scr,NIL);
   Win2 := CreateWindow (45,5,273,15,WIDCMP,WFlags2,NIL,Scr,NIL);
   IF (Win # NIL) AND (Won # NIL) AND (Win2 # NIL) THEN
    IF CreateConsole (Win2^) THEN
      wClrScr (Win2^);
      PutStr (Win2^,ADR(Phrase));
      wSetCursor (Win2^,FALSE);
    IF CreateConsole (Won^) THEN
      wClrScr (Won^);
      PutStr (Won^,ADR("Total"));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Rp := Win^.RPort;
      IF NOT QR THEN
        DrawImage (Win^.RPort^,Icone[0],0,0);
        SetAPen (Rp^,5);
      END;
      IF (CompareString (PhrasePron,"Cobya ah vay voo ra su darjo ?")
              = equal)
      THEN
        DrawImage (Win^.RPort^,Icone[1],0,0);
        SetAPen (Rp^,8);
      ELSIF (CompareString (PhrasePron,"Cobya ah vay voo daypensay ?")
              = equal)
      THEN
        DrawImage (Win^.RPort^,Icone[2],0,0);
        SetAPen (Rp^,2);
      END;

      Move (Rp^,0,0);
      Draw (Rp^,0,255);
      Draw (Rp^,319,255);
      Draw (Rp^,319,0);
      Draw (Rp^,0,0);
      Move (Rp^,1,1);
      Draw (Rp^,1,254);
      Draw (Rp^,318,254);
      Draw (Rp^,318,1);
      Draw (Rp^,1,1);
      Move (Rp^,165,20);
      Draw (Rp^,165,255);
      Move (Rp^,166,20);
      Draw (Rp^,166,255);
      Move (Rp^,43,20);
      Draw (Rp^,319,20);
      Move (Rp^,43,21);
      Draw (Rp^,319,21);
      Move (Rp^,166,237);
      Draw (Rp^,319,237);
      Move (Rp^,166,238);
      Draw (Rp^,319,238);
      IF TableauParametres[1] = 2 THEN
        IF OpenSpeech () THEN
              Res := SayAndReturn (ADR(PhrasePron));
              CopyString (DerPron,PhrasePron);
              CloseSpeech ()
        END;
      END;
      SA := Sactuel;
      DateStamp (DSR);
      NminBEG[IndTemps] := SHORT (DSR.dsMinute);
```

```
                NticksBEG[IndTemps] := SHORT (DSR.dsTick);
                Done := FALSE;
                WHILE (NOT Done) DO
                  Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                  LOOP
                    Msg := GetMsg(Win^.UserPort^);
                    IF (Msg = NIL) THEN EXIT; END;
                    ProcIMsg (Wp, Msg);
                  END;
                END;
                IndTemps := IndTemps - 1;
                IF (( Nbillets > 0) AND (NOT QR) )
                THEN
                   II := 1;
                   WHILE (II <= IndTemps) DO
                      TempsSIBillets := TempsSIBillets + TempsInterm[II];
                      II := II + 1;
                   END;
                   ree := real(TempsSIBillets);
                   ree2 := real (Nbillets);
                   TempsSIBillets := entier (ree / ree2);
                END;
                DeleteConsole (Won^);
              END;
              DeleteConsole (Win2^);
            END;
            EtatSortie := ES;
            Valeur := Somme;
            CloseWindow(Won^);
            CloseWindow(Win2^);
            CloseWindow(Win^);
            END;
            FreeGadgetList (Gl^);
          END;
        CloseScreen(Scr^);
      END;
    END;
  END;
END;
IF Quit = TRUE
THEN
  TabNombre[0] := Nb5000;
  TabNombre[1] := Nb1000;
  TabNombre[2] := Nb500;
  TabNombre[3] := Nb100;
  TabNombre[5] := Nb50;
  TabNombre[6] := Nb20;
  TabNombre[7] := Nb5;
  TabNombre[8] := Nb1;
ELSE
  TabNombreInit[0] := Nb5000;
  TabNombreInit[1] := Nb1000;
  TabNombreInit[2] := Nb500;
  TabNombreInit[3] := Nb100;
  TabNombreInit[5] := Nb50;
  TabNombreInit[6] := Nb20;
  TabNombreInit[7] := Nb5;
  TabNombreInit[8] := Nb1;
END;
END SaisieSommeBillets;

(*  VAR
  p,pp : ARRAY [0..40] OF CHAR;
  q : BOOLEAN;
  v : REAL;
```

```
  e : Str20;

BEGIN

  p := "Combien avez-vous d'argent ?";
  pp := "Cobya ah vay voo darjo ?";
  q := FALSE;
  SaisieSommeBillets (p,pp,q,v,e);      *)

END SSBillets.
```

```
IMPLEMENTATION MODULE PRD;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM Text IMPORT Text;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100,TabNombre,
        TableauTraces,TRecettes,TDepenses,TabNombreInit,TableauParametres,
        Sinit,Max;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Drawing IMPORT SetAPen,Draw,Move;
FROM ModReglette IMPORT Reglette;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;
VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,
                                              postepron,Maxaf : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  For : RealToStringFormat;
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
```

```
  Win,Won,Win2 : WindowPtr;
  Rec,Dep: Image;
  ImgPtr : ImageDescTablePtr;
  ImgCount,Indice, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005,
  IRecettes,IDepenses,Fait,I : INTEGER;
  Somme, SommePrecedente, SommeMax : REAL;
  StrSomme,ES,fonc : Str20;
  TePtr : IntuiTextPtr;
  Prem,Res : BOOLEAN;
  PhrasePron : Str20;
  Rp : RastPortPtr;

PROCEDURE GadgetHandlerPRD (VAR W:Window ; VAR Msg:MsgData ; VAR gad:Gadget);
  BEGIN
    Done := TRUE;
END GadgetHandlerPRD;


PROCEDURE PresentationRecDep;
VAR Total,TotalPrec,ValeurMax : REAL;

  BEGIN

  WITH Wp DO
    procGadgetUp := GadgetHandlerPRD;
  END;

  ValeurMax := Max;
  Scr := CreateScreen (640,256,3,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A52H;
      cmap[04] := 0D80H;
      cmap[05] := 00B6H;
      cmap[06] := 0999H;
      cmap[07] := 00AFH;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 0A52H;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0999H;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
```

```
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),8);

      ImgPtr := FindImageTable(22111111H, ImgCount);
      IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
         WITH Rec DO
            LeftEdge     := 0;
            TopEdge      := 0;
            Width        := ImgPtr^[0].Width;
            Height       := ImgPtr^[0].Height;
            Depth        := ImgPtr^[0].Depth;
            ImageData    := ImgPtr^[0].Data;
            PlanePick    := BYTE(31);
            PlaneOnOff   := BYTE(31);
            NextImage := NIL;
         END;
         WITH Dep DO
            LeftEdge     := 0;
            TopEdge      := 0;
            Width        := ImgPtr^[1].Width;
            Height       := ImgPtr^[1].Height;
            Depth        := ImgPtr^[1].Depth;
            ImageData    := ImgPtr^[1].Data;
            PlanePick    := BYTE(31);
            PlaneOnOff   := BYTE(31);
            NextImage := NIL;
         END;
       END;

BeginGadgetList ();
AddGadgetTextButton (1,1,ADR(" SUITE "));
G1 := EndGadgetList ();

IF (G1 # NIL) THEN
Win2 := CreateWindow (0,0,640,256,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win2 # NIL)
THEN Win := CreateWindow (1,43,638,212,WIDCMP,WFlags2,NIL,Scr,NIL);

IF (Win # NIL) THEN

  DrawImage (Win2^.RPort^,Rec,0,0);
  Rp := Win2^.RPort;
  SetAPen (Rp^,5);
  Move (Rp^,0,0);
  Draw (Rp^,0,255);
  Draw (Rp^,639,255);
  Draw (Rp^,639,0);
  Draw (Rp^,0,0);
  Move (Rp^,300,20);
  Text (Rp^,ADR("RECETTES"),StringLength("RECETTES"));
  Move (Rp^,300,30);
  Text (Rp^,ADR("--------"),StringLength("--------"));
  SetAPen (Rp^,1);

  IF CreateConsole (Win^) THEN
      wClrScr (Win^);
      wSetCursor (Win^,FALSE);
      wMove (Win^,2,7);
      PutStr (Win^,ADR("Il vous restait :"));
      wMove (Win^,51,7);
      ConvRealToString (TRecettes.Tab[0].Valeur,Affichage,2,Decimal);
```

```
        PutStr (Win^,ADR(Affichage));
        Total := TRecettes.Tab[0].Valeur;
        ConvRealToString (Total,Affichage,2,Decimal);
        wMove (Win^,67,7);
        PutStr (Win^,ADR(Affichage));
        TotalPrec := 0.;
        IRecettes := 1;
        Fait := 0;
        Prem := TRUE;
        WHILE ((IRecettes + Fait) <= TRecettes.Indice) DO
          IF (Prem = TRUE)
          THEN
            Prem := FALSE;
            wMove (Win^,2,9);
            PutStr (Win^,ADR("Vous avez reçu :"));
          END;
          wMove (Win^,35,4);
          IF (TableauParametres[6] = 0) THEN
            IF (TableauParametres[9] = 0) THEN PutStr (Win^,ADR("MOMENT"));
                                                wMove (Win^,35,5);
                                                PutStr (Win^,ADR("------"));
                                          ELSE PutStr (Win^,ADR("JOUR"));
                                                wMove (Win^,35,5);
                                                PutStr (Win^,ADR("----"));
            END;
          END;
          wMove (Win^,19,4);
          IF (TableauParametres[7] = 0) THEN PutStr (Win^,ADR("POSTE"));
                                             wMove (Win^,19,5);
                                             PutStr (Win^,ADR("-----"));
          END;
          wMove (Win^,51,4);
          PutStr (Win^,ADR("MONTANT"));
          wMove (Win^,51,5);
          PutStr (Win^,ADR("-------"));
          wMove (Win^,67,4);
          PutStr (Win^,ADR("TOTAL"));
          wMove (Win^,67,5);
          PutStr (Win^,ADR("-----"));
          I := 0;
          WHILE ((IRecettes <= 13) AND (IRecettes + Fait <= TRecettes.Indice-1
DO
            Affichage := " ";
            wMove (Win^,35,9 + I);
            IF (TableauParametres[6] = 0) THEN
               PutStr (Win^,ADR(TRecettes.Tab[IRecettes + Fait].Jour))
            END;
            wMove (Win^,19,9 + I);
            IF (TableauParametres[7] = 0) THEN
               PutStr (Win^,ADR(TRecettes.Tab[IRecettes + Fait].Poste))
            END;

            wMove (Win^,51,9 + I);
            ConvRealToString (TRecettes.Tab[IRecettes+Fait].Valeur,
                                             Affichage,2,Decimal);
            PutStr (Win^,ADR(Affichage));
            TotalPrec := Total;
            Total := Total + TRecettes.Tab[IRecettes + Fait].Valeur;
            ConvRealToString (Total,Affichage,2,Decimal);
            wMove (Win^,67,9 + I);
            PutStr (Win^,ADR(Affichage));

            IRecettes := IRecettes + 1;
            I := I + 1;
          END (* WHILE1 *);
          Won := CreateWindow (280,230,200,24,WIDCMP,WFlags2,Gl,Scr,NIL);
```

```
        IF Won # NIL THEN
        Done := FALSE;
        WHILE NOT Done DO
          Sig := Wait (SignalSet{CARDINAL (Won^.UserPort^.mpSigBit)});
          LOOP
            Msg := GetMsg (Won^.UserPort^);
            IF (Msg = NIL) THEN EXIT; END;
            ProcIMsg (Wp,Msg);
          END;
        END;
        CloseWindow (Won^);
        IRecettes := 0;
        Fait := Fait + 14;
        wClrScr (Win^);

        END;
      END (* WHILE2 *);

  DeleteConsole (Win^);
  END (* IF CreateConsole *);

CloseWindow (Win^);
END;
CloseWindow (Win2^);
END;



IF TableauParametres[11] = 1 THEN
Reglette ("R",FALSE);
END;

Win2 := CreateWindow (0,0,640,256,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win2 # NIL) THEN
Win := CreateWindow (1,43,638,212,WIDCMP,WFlags2,NIL,Scr,NIL);


IF (Win # NIL) THEN

  DrawImage (Win2^.RPort^,Dep,0,0);
  Rp := Win2^.RPort;
  SetAPen (Rp^,2);
  Move (Rp^,0,0);
  Draw (Rp^,0,255);
  Draw (Rp^,639,255);
  Draw (Rp^,639,0);
  Draw (Rp^,0,0);
  Move (Rp^,300,20);
  Text (Rp^,ADR("DEPENSES"),StringLength("DEPENSES"));
  Move (Rp^,300,30);
  Text (Rp^,ADR("--------"),StringLength("--------"));
  SetAPen (Rp^,1);
  IF CreateConsole (Win^) THEN
    wClrScr (Win^);
    wSetCursor (Win^,FALSE);
    IDepenses := 0;
    Fait := 0;
    WHILE ((IDepenses + Fait) <= TDepenses.Indice) DO
        wMove (Win^,28,4);
        IF (TableauParametres[6] = 0) THEN
          IF (TableauParametres[9] = 0) THEN PutStr (Win^,ADR("MOMENT"));
                                                wMove(Win^,28,5);
                                                PutStr (Win^,ADR("------"));
                                            ELSE PutStr (Win^,ADR("JOUR"));
                                                wMove(Win^,28,5);
                                                PutStr (Win^,ADR("----"));
```

```
            END;
          END;
          wMove (Win^,12,4);
          IF (TableauParametres[7] = 0) THEN PutStr (Win^,ADR("POSTE"));
                                             wMove (Win^,12,5);
                                             PutStr (Win^,ADR("-----"));
          END;
          wMove (Win^,44,4);
          PutStr (Win^,ADR("MONTANT"));
          wMove (Win^,44,5);
          PutStr (Win^,ADR("-------"));
          wMove (Win^,60,4);
          PutStr (Win^,ADR("TOTAL"));
          wMove (Win^,60,5);
          PutStr (Win^,ADR("-----"));
          I := 0;
          WHILE ((IDepenses <= 14) AND (IDepenses + Fait <= TDepenses.Indice-1
DO
            Affichage := " ";
            wMove (Win^,28,7 + I);
            IF (TableauParametres[6] = 0) THEN
               PutStr (Win^,ADR(TDepenses.Tab[IDepenses + Fait].Jour))
            END;
            wMove (Win^,12,7 + I);
            IF (TableauParametres[7] = 0) THEN
               PutStr (Win^,ADR(TDepenses.Tab[IDepenses + Fait].Poste))
            END;
            wMove (Win^,44,7 + I);
            ConvRealToString (TDepenses.Tab[IDepenses + Fait].Valeur,
                                             Affichage,2,Decimal);
            PutStr (Win^,ADR(Affichage));
            TotalPrec := Total;
            Total := Total - TDepenses.Tab[IDepenses + Fait].Valeur;
            ConvRealToString (Total,Affichage,2,Decimal);
            wMove (Win^,60,7 + I);
            PutStr (Win^,ADR(Affichage));
            IDepenses := IDepenses + 1;
            I := I + 1;
           END (* WHILE1 *);
          Won := CreateWindow (280,230,200,24,WIDCMP,WFlags2,Gl,Scr,NIL);
          Done := FALSE;
          WHILE NOT Done DO
            Sig := Wait (SignalSet{CARDINAL (Won^.UserPort^.mpSigBit)});
            LOOP
              Msg := GetMsg (Won^.UserPort^);
              IF (Msg = NIL) THEN EXIT; END;
              ProcIMsg (Wp,Msg);
            END;
          END;
          CloseWindow (Won^);
          IDepenses := 0;
          Fait := Fait + 15;
          wClrScr (Win^);
        END (* WHILE2 *);

    DeleteConsole (Win^);
    END (* IF CreateConsole *);

 CloseWindow (Win^);
 END;
 CloseWindow (Win2^);
 END;
FreeGadgetList (Gl^);
END (* IF (Gl # NIL) *);
CloseScreen (Scr^);
END;
```

```
IF TableauParametres[11] = 1 THEN
Reglette ("D",FALSE);
END;
END PresentationRecDep;

(*   BEGIN

   PresentationRecDep;  *)

END PRD.
```

```
IMPLEMENTATION MODULE ModNom;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM MathLib0 IMPORT entier,real;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr,ScreenToFront;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenInputFile,
                  CloseInput,Done;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString,ConvStringToUpperCase;
FROM Conversions IMPORT ConvNumberToString;
FROM VariablesGlobales IMPORT Nom, Str3, Str9,Str20,Str40, TabNombre,
   TableauTraces,TRecettes,TDepenses,TabNombreInit,TableauParametres,Max,Str10
   TableauChaine,AncMinute,AncTick,TableauTemps ;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                            ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done1,DoneOK,OK,OKNombre,Done2 : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  For : RealToStringFormat;
  Scr : ScreenPtr;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Wun : WindowPtr;
  TabImages: ARRAY [0..49] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  FileName : ARRAY [0..40] OF CHAR;
  Ms : MenuPtr;
  G1,G12 : GadgetPtr;
```

```
  Wp,Wp2 : WindowProc;
  Req,Req2 : Requester;
  Sig : SignalSet;
  Msg, Msg2 : IntuiMessagePtr;
  Somme, SommePrecedente, SommeMax : REAL;
  StrSomme,ES,fonc : Str20;
  Res,CWB : BOOLEAN;
  PhrasePron : Str20;
  JJourStr,ANStr : Str20;
  DSR : DateStampRecord;
  MOIS,MOISPREC,Y : INTEGER;
  TrouveMois,TrouveAn : BOOLEAN;
  NBJours : ARRAY [0..11] OF CARDINAL;
  NomMois : ARRAY [0..11] OF Str9;
  Re,Re2,an : REAL;
  X : CARDINAL;
  Quatre : REAL;
  En : INTEGER;
  JJour,AN : INTEGER;
  ActMinute,ActTick,ActTick2,ent : INTEGER;
  ree : REAL;

PROCEDURE EnleveBlancs (VAR S : Str20);
  VAR Indice1,Indice2 : CARDINAL;
      S2 : Str20;
  BEGIN
    Indice1 := 0;
    Indice2 := 0;
    WHILE (Indice1 <= (StringLength (S) - 1)) DO
      IF (S[Indice1] # " ") AND (S[Indice1] # "-") AND (S[Indice1] # "_")
      THEN
        IF (S[Indice1] = "é") OR (S[Indice1] = "è")
        THEN S2[Indice2] := "E";
             Indice2 := Indice2 + 1;
        ELSE
           S2[Indice2] := S[Indice1];
           Indice2 := Indice2 + 1;
        END;
      END (* IF *);
      Indice1 := Indice1 + 1;
    END (* WHILE *);
    S2[Indice2] := S[Indice1];
    CopyString (S,S2);
  END EnleveBlancs;

PROCEDURE GadgetHandlerNO (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF
      0 : |
      1 : Done1 := TRUE; Efface := FALSE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab [TableauChaine.Indice] := 2;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
```

```
            TableauChaine.Indice := TableauChaine.Indice + 1;
        END;
        CopyString (Trace,"Il confirme le nom");
        IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
    2 : Done1 :=TRUE;
        IF TableauChaine.Indice <= 499
        THEN TableauChaine.Tab [TableauChaine.Indice] := 2;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
             TableauChaine.Tab [TableauChaine.Indice] := 0;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
        END;

        CopyString (Trace,"Il efface le nom");
        IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END
  END;
END GadgetHandlerNO;

PROCEDURE GadgetHandler2 (VAR w: Window; VAR Msg2 : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF
      0 : ;
      1 : ;
      2 : Done2 :=TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab [TableauChaine.Indice] := 0;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
```

```
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;

  END;
END GadgetHandler2;


(*****************************************************************************)
(*                                SaisieNom                                  *)
(*****************************************************************************)

PROCEDURE SaisieNom;


TYPE
  TabPtr = POINTER TO ARRAY [0..7] OF INTEGER;

BEGIN
(*     IF OpenSpeech () THEN *)

    TableauChaine.Indice :=0;
    TableauChaine.Tab [TableauChaine.Indice] := 0;
    TableauTemps [TableauChaine.Indice] := 0;
    DateStamp(DSR);
    AncMinute := SHORT(DSR.dsMinute);
    AncTick := SHORT(DSR.dsTick);
    TableauChaine.Indice := TableauChaine.Indice + 1;

     WITH Wp DO
      procGadgetUp := GadgetHandlerNO;
    END;
     WITH Wp2 DO
      procGadgetUp := GadgetHandler2;
    END;

    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
```

```
    cmap[26] := 0999H;
    cmap[27] := 0AAAH;
    cmap[28] := 0CCCH;
    cmap[29] := 0DDDH;
    cmap[30] := 0EEEH;
    cmap[31] := 0FFFH;
    LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);


DateStamp (DSR);
X := SHORT (DSR.dsDays);
Y := X;
X := X + 1;
AN := 78;
TrouveAn := FALSE;
NBJours [0] := 31;
NBJours [2] := 31;
NBJours [3] := 30;
NBJours [4] := 31;
NBJours [5] := 30;
NBJours [6] := 31;
NBJours [7] := 31;
NBJours [8] := 30;
NBJours [9] := 31;
NBJours [10] := 30;
NBJours [11] := 31;
NomMois [0] := "Janvier";
NomMois [1] := "Février";
NomMois [2] := "Mars";
NomMois [3] := "Avril";
NomMois [4] := "Mai";
NomMois [5] := "Juin";
NomMois [6] := "Juillet";
NomMois [7] := "Aout";
NomMois [8] := "Septembre";
NomMois [9] := "Octobre";
NomMois [10] := "Novembre";
NomMois [11] := "Décembre";
WHILE NOT TrouveAn DO
    an := real (AN);
    Quatre := 4.0;
    Re := an/Quatre;
    En := entier (Re);
    Re2 := real (En);
    IF Re=Re2
    THEN
        IF X>366
        THEN
            X := X - 366;
            AN := AN + 1;
        ELSE
            NBJours [1] := 29;
            TrouveAn := TRUE;
        END;
    ELSE
        IF X>365
        THEN
            X := X-365;
            AN := AN+1;
        ELSE
            NBJours [1]:=28;
            TrouveAn := TRUE;
        END;
    END;
END;
TrouveMois := FALSE;
```

```
MOISPREC := 11;
MOIS := 0;
WHILE NOT TrouveMois DO
   IF  X > NBJours [MOIS]
   THEN X := X - NBJours [MOIS];
        MOISPREC := MOIS;
        MOIS := MOIS +1;
   ELSE TrouveMois := TRUE;
   END;
END;
JJour := X;
CopyString (Trace,"Date de la session : ");
ConvNumberToString (JJourStr,LONGINT(JJour),FALSE,10,2," ");
ConcatString (Trace,JJourStr);
ConcatString (Trace," ");
ConcatString (Trace,NomMois [MOIS]);
ConcatString (Trace," ");
ConvNumberToString (ANStr,LONGINT(AN),FALSE,10,2," ");
ConcatString (Trace,ANStr);
IF (TableauTraces.Indice <= 499) THEN
  CopyString (TableauTraces.Tab[TableauTraces.Indice],Trace);
  TableauTraces.Indice := TableauTraces.Indice + 1;
END;

  Win := CreateWindow (0,0,320,256,WIDCMP,WFlags,NIL,Scr,NIL);
  Won := CreateWindow (70,40,180,60,WIDCMP,WFlags2,NIL,Scr,NIL);
  IF (Won # NIL) THEN
    Efface := TRUE;
    WHILE Efface = TRUE DO
      ActivateWindow (Won^);
      BeginGadgetList();
      GadgetTypeReq := TRUE;
      GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                       RelVerify});
      AddGadgetTextButton (60,15,ADR("EST-CE BIEN JUSTE ?"));
      GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
      AddGadgetTextButton (90,35,ADR("OUI"));
      AddGadgetTextButton (170,35,ADR("NON"));
      G1 := EndGadgetList();
      InitRequester (Req);
      WITH Req DO
        OlderRequest := NIL;
        LeftEdge := 10;
        TopEdge := 100;
        Width := 300;
        Height := 60;
        ReqGadget := G1;
        ReqBorder := NIL;
        ReqText := NIL;
        Flags := RequesterFlagsSet {};
        BackFill := BYTE(10);
        ReqLayer := NIL;
        ImageBMap := NIL;
      END;

      BeginGadgetList();
      GadgetTypeReq := TRUE;
      GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                       RelVerify});
      AddGadgetTextButton (30,10,ADR("JE NE TROUVE PAS VOTRE NOM"));
      GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
      AddGadgetTextButton (80,25,ADR("ESSAYEZ ENCORE"));
      GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
```

```
        AddGadgetTextButton (120,45,ADR( OK ));
        G12 := EndGadgetList();
        InitRequester (Req2);
        WITH Req2 DO
          OlderRequest := NIL;
          LeftEdge := 10;
          TopEdge := 100;
          Width := 300;
          Height := 60;
          ReqGadget := G12;
          ReqBorder := NIL;
          ReqText := NIL;
          Flags := RequesterFlagsSet {};
          BackFill := BYTE(3);
          ReqLayer := NIL;
          ImageBMap := NIL;
        END;

           IF CreateConsole (Won^) THEN
            wMove (Won^,8,1);PutStr(Won^,ADR("BONJOUR !"));
            wMove (Won^,2,1);PutStr(Won^,ADR("QUEL EST VOTRE NOM ?"));
            wMove (Won^,1,4);
            IF (TableauParametres[1] = 2) THEN
(*              Res := SayAndReturn (ADR("Bonjoor, keyl hey votr noh ?"));
      *)      END;
            CopyString (Nom,"");
            GetStr (Won^,ADR (Nom), SIZE (Nom));
            IF CompareString (Nom,"") # equal THEN EnleveBlancs (Nom); END;
            ConvStringToUpperCase (Nom);
            CopyString (Trace,"Il donne le nom : ");
            ConcatString (Trace,Nom);
            IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END;
            wSetCursor (Won^,FALSE);
            IF Request (Req, Win^) THEN
            IF (TableauParametres[1] = 0) THEN
(*              Res := SayAndReturn (ADR("Sy Asay just, tpay sur fyny ?"));
              Res := SayAndReturn (ADR("Synon, tpay sur effahs ?"));
     *)       END;
            Done1 := FALSE;
            WHILE (NOT Done1) DO
              Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
              LOOP
                Msg := GetMsg(Win^.UserPort^);
                IF (Msg = NIL) THEN EXIT; END;
                ProcIMsg (Wp, Msg);
              END;
            END;
            IF Efface = FALSE (* càd il confirme le nom *)
               THEN FileName := "Param:";
                    ConcatString (FileName,Nom);
                    ConcatString (FileName,".MAX");
                    OpenInputFile (FileName);
                    IF NOT Done THEN
                            Efface := TRUE;
                            ScreenToFront (Scr^);
                            IF Request (Req2, Win^) THEN
                            Done2 := FALSE;
                            WHILE (NOT Done2) DO
                            Sig := Wait(SignalSet
                               {CARDINAL(Win^.UserPort^.mpSigBit)});
                              LOOP
                                Msg2 := GetMsg(Win^.UserPort^);
                                IF (Msg2 = NIL) THEN EXIT; END;
```

```
                                ProcIMsg (Wp2, Msg2);
                              END;
                            END;
                           ELSE WriteString ("requester pas créé");END;
                          ELSE CloseInput;
                      END;
                    END; (* if efface *)
                  ELSE WriteString ("requester pas créé");END;
             DeleteConsole (Won^);
             END;

           FreeGadgetList (G1^);
           FreeGadgetList (G12^);
           END; (* while efface *)
       CloseWindow(Won^);
       END;
       CloseWindow(Win^);
       CloseScreen(Scr^);
     END;

(* CloseSpeech ();
END; *)
END SaisieNom;

(*  BEGIN
    SaisieNom;    *)

END ModNom.
```

```
IMPLEMENTATION MODULE ModPosteRec;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM MathLib0 IMPORT entier,real;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40, TabNombre,TableauTraces,
        TRecettes,TDepenses,TabNombreInit,TableauParametres,Max,Str100,
        TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  For : RealToStringFormat;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Wun : WindowPtr;
  TabImages: ARRAY [0..3] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
```

```
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005,
  IRecettes,IDepenses,Fait,I : INTEGER;
  Somme, SommePrecedente, SommeMax : REAL;
  StrSomme,ES,fonc : Str20;
  TePtr : IntuiTextPtr;
  Res : BOOLEAN;
  PhrasePron : Str100;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree : REAL;
  DSR : DateStampRecord;
  Rp : RastPortPtr;

PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : OK := TRUE ; DoneOK := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab[TableauChaine.Indice] :=9;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real (ent);
                    TableauTemps [TableauChaine.Indice] := entier (ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           END;
           Trace := "Il confirme ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END; |
     1 : OK := FALSE; DoneOK := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab[TableauChaine.Indice] :=9;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real (ent);
                    TableauTemps [TableauChaine.Indice] := entier (ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
                TableauChaine.Tab[TableauChaine.Indice] :=8;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
```

```
                    ent := ActTick - AncTick;
                    ree := real (ent);
                    TableauTemps [TableauChaine.Indice] := entier (ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il infirme ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
    2 :
  END;
END GadgetHandlerOK;

PROCEDURE OKRequesterPO (VAR P : Str20;VAR PP : Str20);
BEGIN
            IF OpenSpeech() THEN
            Confirmation := "C'est bien ";
            ConcatString (Confirmation,P);
            ConcatString (Confirmation," ?");
            BeginGadgetList ();
            GadgetTypeReq := TRUE;
            GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                             RelVerify});
            AddGadgetTextButton (10,45, ADR("OUI"));
            AddGadgetTextButton (230,45, ADR("NON"));
            AddGadgetTextButton (5,15, ADR(Confirmation));
            GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
            GlOK := EndGadgetList ();
            InitRequester (ReqOK);
            WITH ReqOK DO
                   OlderRequest := NIL;
                   LeftEdge := 30;
                   TopEdge := 160;
                   Width := 270;
                   Height := 60;
                   ReqGadget := GlOK;
                   ReqBorder := NIL;
                   ReqText := NIL;
                   Flags := RequesterFlagsSet {};
                   BackFill := BYTE(10);
                   ReqLayer := NIL;
                   ImageBMap := NIL;
            END;
            IF Request (ReqOK, Win^) THEN
               IF (TableauParametres[1] = 2) THEN
                 CopyString (PhrasePron,"say bien ");
                 ConcatString (PhrasePron,PP);
                 Res := SayAndReturn (ADR(PhrasePron));
               END;
               DoneOK := FALSE;
               WHILE NOT DoneOK DO
                   Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                   LOOP
                      MsgOK := GetMsg (Win^.UserPort^);
                      IF (MsgOK = NIL) THEN EXIT; END;
                      ProcIMsg (WpOK, MsgOK);
                   END;
               END;
            END;
            FreeGadgetList (GlOK^);
CloseSpeech();
END;
```

```
END OKRequesterPO;

(*****************************************************************************)
(*                              GadgetHandlers                               *)
(*****************************************************************************)

PROCEDURE GadgetHandlerPO (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR P : StringInfoPtr;
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
BEGIN
  CASE Gad.GadgetID OF
        0 : poste := "les parents";
            postepron := "lay pahroh ?";
           Trace := "Il appuie sur parents ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Parents" END |
        1 : poste := "l'argent de poche";
            postepron := "larjo dir posh ?";
           Trace := "Il appuie sur argent de poche ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Poche" END |
        2 : poste := "du travail";
            postepron := "la trahvi ?";
           Trace := "Il appuie sur travail ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Travail" END |
        3 : poste := "la tirelire";
            postepron := "lah teer leer ?";
           Trace := "Il appuie sur tirelire ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Tirelire" END |
        4 :
  END;
END GadgetHandlerPO;


(*****************************************************************************)
(*                                PosteRec                                   *)
(*****************************************************************************)

PROCEDURE PosteRec (VAR ValeurPoste : Str20);


BEGIN

    IF OpenSpeech () THEN END;
    WITH Wp DO
      procGadgetUp := GadgetHandlerPO;
    END;
    WITH WpOK DO
```

```
  procGadgetUp := GadgetHandlerOK;
END;
Scr := CreateScreen (320,256,5,NIL);
IF (Scr # NIL) THEN
  cmap[00] := 0000H;
  cmap[01] := 0FFFH;
  cmap[02] := 0E00H;
  cmap[03] := 0A00H;
  cmap[04] := 0D80H;
  cmap[05] := 0FE0H;
  cmap[06] := 08F0H;
  cmap[07] := 0080H;
  cmap[08] := 00B6H;
  cmap[09] := 00DDH;
  cmap[10] := 00AFH;
  cmap[11] := 007CH;
  cmap[12] := 000FH;
  cmap[13] := 070FH;
  cmap[14] := 0C0EH;
  cmap[15] := 0C08H;
  cmap[16] := 0620H;
  cmap[17] := 0E52H;
  cmap[18] := 0A52H;
  cmap[19] := 0FCAH;
  cmap[20] := 0333H;
  cmap[21] := 0444H;
  cmap[22] := 0555H;
  cmap[23] := 0666H;
  cmap[24] := 0777H;
  cmap[25] := 0888H;
  cmap[26] := 0999H;
  cmap[27] := 0AAAH;
  cmap[28] := 0CCCH;
  cmap[29] := 0DDDH;
  cmap[30] := 0EEEH;
  cmap[31] := 0FFFH;
  LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);
  ImgPtr := FindImageTable(11111111H, ImgCount);
  IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
    Indice := 0;
    WHILE (Indice <= 3) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr^[Indice+8].Width;
        Height      := ImgPtr^[Indice+8].Height;
        Depth       := ImgPtr^[Indice+8].Depth;
        ImageData   := ImgPtr^[Indice+8].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice := Indice + 1;
    END;
     Confirmation := "C'est ";
     BeginGadgetList();
     AddGadgetImageButton (20,40,TabImages[0]);
     AddGadgetImageButton (88,40,TabImages[1]);
     AddGadgetImageButton (156,40,TabImages[2]);
     AddGadgetImageButton (224,40,TabImages[3]);
     AddGadgetTextButton (50,20,
                          ADR("D'où vient cette recette ?"));
     Gl := EndGadgetList();
     IF (Gl # NIL) THEN
        Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,Gl,Scr,NIL);
        IF (Win # NIL)  THEN
```

```
                Rp := Win^.RPort;
                SetAPen (Rp^,8);
                Move (Rp^,0,0);
                Draw (Rp^,0,255);
                Draw (Rp^,319,255);
                Draw (Rp^,319,0);
                Draw (Rp^,0,0);
                SetAPen (Rp^,1);
                IF (TableauParametres[1] = 2) THEN
                   Res := SayAndReturn (ADR("doo vien set ra set ?"));
                END;
                 CloseSpeech ();
                Done := FALSE;
                WHILE (NOT Done) DO
                  Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                  LOOP
                    Msg := GetMsg(Win^.UserPort^);
                    IF (Msg = NIL) THEN EXIT; END;
                    ProcIMsg (Wp, Msg);
                  END;
                END;
              ValeurPoste := poste;
              CloseWindow(Win^);
              END;
              FreeGadgetList (Gl^);
            END;
          CloseScreen(Scr^);
        END;
    END;

END PosteRec;

(*   VAR v : Str20;

BEGIN

    PosteRec (v);     *)

END ModPosteRec.
```

```
IMPLEMENTATION  MODULE ModMois;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM MathLib0 IMPORT entier,real;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT CopyString, CompareString, Relation,ConcatString,
       StringLength,LocateChar,ExtractSubString;
FROM Conversions IMPORT ConvNumberToString;
FROM VariablesGlobales IMPORT Nom,Str3,Str9,Str20,Str40,Str60,Str100,
     TabNombre,TableauTraces,TRecettes,TDepenses,TabNombreInit,
     TableauParametres,Max, TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                            ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";

VAR
  moment,poste,Nombre,momentpron,postepron,VJ,Trace2 : Str20;
  jourpron,jour,JourPron2,Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  For : RealToStringFormat;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Wun : WindowPtr;
  TabAffich,TabJoursVerts,TabJoursBlancs,
            TabJoursRouges : ARRAY [0..30] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice, i,jourdanssemaine,X,
       indicejour,IndiceBlanc,JourChoisi,jourselectionne : CARDINAL;
  Y : INTEGER;
```

```
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005,
  IRecettes,IDepenses,Fait,I,JJour : INTEGER;
  AN,MOIS,MOISPREC,En : CARDINAL;
  TrouveMois,TrouveAn : BOOLEAN;
  Re,Re2,an : REAL;
  NBJours : ARRAY [0..11] OF CARDINAL;
  NomMois : ARRAY [0..11] OF Str9;
  MoisChoisi,MoisPron,MoisPron2 : Str9;
  Somme, SommePrecedente, SommeMax : REAL;
  StrSomme,ES,fonc : Str20;
  TePtr : IntuiTextPtr;
  Res : BOOLEAN;
  PhrasePron : Str40;
  Quatre : REAL;
  ABS,ORD,jourdesem: CARDINAL;
  N : INTEGER;
  jchoisi: LONGINT;
  Mo : CHAR;
  DSR : DateStampRecord;
  ree : REAL;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  Rp : RastPortPtr;

PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : OK := TRUE ; DoneOK := TRUE;
           IF TableauChaine.Indice <= 499
           THEN IF Mo = "D"
                THEN TableauChaine.Tab[TableauChaine.Indice] :=19;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
                     ActTick := SHORT(DSR.dsTick);
                     ActTick2 := ActTick;
                     IF ActMinute > AncMinute
                     THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                     END;
                     ent := ActTick - AncTick;
                     ree := real(ent);
                     TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                     AncMinute := ActMinute;
                     AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE TableauChaine.Tab[TableauChaine.Indice] :=11;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
                     ActTick := SHORT(DSR.dsTick);
                     ActTick2 := ActTick;
                     IF ActMinute > AncMinute
                     THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                     END;
                     ent := ActTick - AncTick;
                     ree := real(ent);
                     TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                     AncMinute := ActMinute;
                     AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
```

```
            END;
        END;
        Trace := "Il confirme ";
        IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END; !
1 : OK := FALSE; DoneOK := TRUE;
        IF TableauChaine.Indice <= 499
        THEN IF Mo = "D"
            THEN TableauChaine.Tab[TableauChaine.Indice] :=19;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
                 TableauChaine.Tab[TableauChaine.Indice] :=18;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
            ELSE TableauChaine.Tab[TableauChaine.Indice] :=11;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
                 TableauChaine.Tab[TableauChaine.Indice] :=10;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
```

```
                END;
            END;
            Trace := "Il infirme ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
      2 :
   END;
END GadgetHandlerOK;


PROCEDURE OKRequesterMO (VAR J : Str40;VAR JP : Str40);
BEGIN
            IF OpenSpeech () THEN
            Confirmation := "C'est bien le ";
            ConcatString (Confirmation,J);
            ConcatString (Confirmation," ?");
            BeginGadgetList ();
            GadgetTypeReq := TRUE;
            GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                             RelVerify});
            AddGadgetTextButton (10,35, ADR("OUI"));
            AddGadgetTextButton (260,35, ADR("NON"));
            AddGadgetTextButton (5,15, ADR(Confirmation));
            GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
            G1OK := EndGadgetList ();
            InitRequester (ReqOK);
            WITH ReqOK DO
                   OlderRequest := NIL;
                   LeftEdge := 10;
                   TopEdge := 180;
                   Width := 300;
                   Height := 50;
                   ReqGadget := G1OK;
                   ReqBorder := NIL;
                   ReqText := NIL;
                   Flags := RequesterFlagsSet {};
                   BackFill := BYTE(10);
                   ReqLayer := NIL;
                   ImageBMap := NIL;
            END;
            IF Request (ReqOK, Win^) THEN
               IF (TableauParametres[1] = 2) THEN
                  CopyString (PhrasePron,"Say Bien la ");
                  ConcatString (PhrasePron,JP);
                  Res := SayAndReturn (ADR(PhrasePron));
               END;
               DoneOK := FALSE;
               WHILE NOT DoneOK DO
                    Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                    LOOP
                       MsgOK := GetMsg (Win^.UserPort^);
                       IF (MsgOK = NIL) THEN EXIT; END;
                       ProcIMsg (WpOK, MsgOK);
                    END;
               END;
            END;
            FreeGadgetList (G1OK^);
CloseSpeech ();
END;
END OKRequesterMO;

PROCEDURE GadgetHandlerMO ( VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget
VAR P : StringInfoPtr;
```

```
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;

BEGIN
  IF (Gad.GadgetID <> 0) AND (Gad.GadgetID <= IndiceBlanc)
  THEN
    IF Gad.GadgetID > NBJours [MOISPREC]
    THEN JourChoisi := (Gad.GadgetID - NBJours [MOISPREC]);
         MoisChoisi := NomMois [MOIS];
    ELSE JourChoisi := Gad.GadgetID;
         MoisChoisi := NomMois [MOISPREC];
    END;
    jchoisi := JourChoisi;
    CopyString (Trace2,"");
    ConvNumberToString (Trace2,jchoisi,FALSE,10,2," ");
    ConcatString (Trace2," ");
    ConcatString (Trace2,MoisChoisi);
    MoisPron := MoisChoisi;
    CASE JourChoisi OF
        0 : JourPron2 := "" |
        1 : JourPron2 := "an" |
        2 : JourPron2 := "duh" |
        3 : JourPron2 := "trwa" |
        4 : JourPron2 := "katr" |
        5 : JourPron2 := "senk" |
        6 : JourPron2 := "sys" |
        7 : JourPron2 := "set" |
        8 : JourPron2 := "weet" |
        9 : JourPron2 := "nuf" |
       10 : JourPron2 := "dhe" |
       11 : JourPron2 := "onz" |
       12 : JourPron2 := "dooz" |
       13 : JourPron2 := "trayze" |
       14 : JourPron2 := "katorz" |
       15 : JourPron2 := "kenz" |
       16 : JourPron2 := "says" |
       17 : JourPron2 := "dys set" |
       18 : JourPron2 := "dys weet" |
       19 : JourPron2 := "dys nuf" |
       20 : JourPron2 := "vant" |
       21 : JourPron2 := "vant an" |
       22 : JourPron2 := "vant duh" |
       23 : JourPron2 := "vant trwa" |
       24 : JourPron2 := "vant katr" |
       25 : JourPron2 := "vant senk" |
       26 : JourPron2 := "vant sys" |
       27 : JourPron2 := "vant set" |
       28 : JourPron2 := "vant weet" |
       29 : JourPron2 := "vant nuf" |
       30 : JourPron2 := "tran't" |
       31 : JourPron2 := "tran't an"
    END;
    ConcatString (JourPron2,MoisPron);
    jourselectionne := indicejour + Gad.GadgetID - 1;
    jourdanssemaine := ((jourselectionne - 1) MOD 7);
    CASE jourdanssemaine OF
        0 :   jour := "Lundi ";
              jourpron := "lundy";
              Trace := "Il appuie sur lundi ";
              ConcatString (Trace,Trace2);
              IF TableauTraces.Indice <= 499 THEN
                  CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace
                  TableauTraces.Indice :=  TableauTraces.Indice + 1;
              END;
              ConcatString (jour,Trace2);
              ConcatString (jourpron,JourPron2);
              OKRequesterMO (jour,jourpron);
```

```
        IF OK THEN Done := TRUE END; |
1 :     jour := "Mardi ";
        jourpron := "mardy";
        Trace := "Il appuie sur mardi ";
        ConcatString (Trace,Trace2);
        IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
        ConcatString (jour,Trace2);
        ConcatString (jourpron,JourPron2);
        OKRequesterMO (jour,jourpron);
        IF OK THEN Done := TRUE END; |
2 :     jour := "Mercredi ";
        jourpron := "merkrdy";
        Trace := "Il appuie sur mercredi ";
        ConcatString (Trace,Trace2);
        IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
        ConcatString (jour,Trace2);
        ConcatString (jourpron,JourPron2);
        OKRequesterMO (jour,jourpron);
        IF OK THEN Done := TRUE END; |
3 :     jour := "Jeudi ";
        jourpron := "judy";
        Trace := "Il appuie sur jeudi ";
        ConcatString (Trace,Trace2);
        IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
        ConcatString (jour,Trace2);
        ConcatString (jourpron,JourPron2);
        OKRequesterMO (jour,jourpron);
        IF OK THEN Done := TRUE END; |
4 :     jour := "Vendredi ";
        jourpron := "vendrdy";
        Trace := "Il appuie sur vendredi ";
        ConcatString (Trace,Trace2);
        IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
        ConcatString (jour,Trace2);
        ConcatString (jourpron,JourPron2);
        OKRequesterMO (jour,jourpron);
        IF OK THEN Done := TRUE END; |
5 :     jour := "Samedi ";
        jourpron := "samdy";
        Trace := "Il appuie sur samedi ";
        ConcatString (Trace,Trace2);
        IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
        ConcatString (jour,Trace2);
        ConcatString (jourpron,JourPron2);
        OKRequesterMO (jour,jourpron);
        IF OK THEN Done := TRUE END; |
6 :     jour := "Dimanche ";
        jourpron := "demansh";
        Trace := "Il appuie sur dimanche ";
        ConcatString (Trace,Trace2);
        IF TableauTraces.Indice <= 499 THEN
```

```
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
              END;
              ConcatString (jour,Trace2);
              ConcatString (jourpron,JourPron2);
              OKRequesterMO (jour,jourpron);
              IF OK THEN Done := TRUE END;
            END;
    END;
END GadgetHandlerMO;

(*****************************************************************************)
(*                                  Mois                                     *)
(*****************************************************************************)


PROCEDURE Mois ( VAR ValeurJour : Str20; Mouvement : CHAR);

VAR ImgPtr,ImgPtr2,ImgPtr3,ImgPtr4,ImgPtr5 : ImageDescTablePtr;
    ImgCount,ImgCount2,ImgCount3,ImgCount4,ImgCount5 : CARDINAL;
    i,I,J,k,l,m : INTEGER;
    Indice1,Indice2,LI1,LI2 : LONGINT;
    DSR : DateStampRecord;
    TabMois : ARRAY [0..11] OF Image;
    TabJours : ARRAY [0..6] OF Image;

BEGIN

    Mo := Mouvement;
    IF OpenSpeech () THEN END;
    WITH Wp DO
      procGadgetUp := GadgetHandlerMO;
    END;
    WITH WpOK DO
      procGadgetUp := GadgetHandlerOK;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
```

```
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

  i := 0;
  ImgPtr := FindImageTable (51111111H,ImgCount);
  IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
    WHILE (i <= 30) DO
      WITH TabJoursVerts[i] DO
        LeftEdge   := 0;
        TopEdge    := 0;
        Width      := ImgPtr^[i].Width;
        Height     := ImgPtr^[i].Height;
        Depth      := ImgPtr^[i].Depth;
        ImageData  := ImgPtr^[i].Data;
        PlanePick  := BYTE (31);
        PlaneOnOff := BYTE (31);
        NextImage  := NIL;
      END;
      i := i + 1;
    END;
  END;
   i := 0;
   ImgPtr2 := FindImageTable (52222222H,ImgCount2);
   IF (ImgPtr2 # NIL ) AND (ImgCount2 # 0) THEN
     WHILE (i <=  30 ) DO
      WITH TabJoursRouges[i] DO
        LeftEdge   := 0;
        TopEdge    := 0;
        Width      := ImgPtr2^[i].Width;
        Height     := ImgPtr2^[i].Height;
        Depth      := ImgPtr2^[i].Depth;
        ImageData  := ImgPtr2^[i].Data;
        PlanePick  := BYTE (31);
        PlaneOnOff := BYTE (31);
        NextImage  := NIL;
      END;
      i := i + 1;
    END;
 END;
   i:=0;
   ImgPtr3 := FindImageTable (53333333H,ImgCount3);
   IF (ImgPtr3 # NIL) AND (ImgCount3 # 0) THEN
    WHILE (i <= 30) DO
      WITH TabJoursBlancs[i] DO
        LeftEdge   := 0;
        TopEdge    := 0;
        Width      := ImgPtr3^[i].Width;
        Height     := ImgPtr3^[i].Height;
        Depth      := ImgPtr3^[i].Depth;
        ImageData  := ImgPtr3^[i].Data;
        PlanePick  := BYTE (31);
        PlaneOnOff := BYTE (31);
        NextImage  := NIL;
      END;
      i := i + 1;
   END;
END;
  i := 0;
  ImgPtr4 := FindImageTable (54444444H,ImgCount4);
  IF (ImgPtr4 # NIL) AND (ImgCount4 # 0) THEN
    WHILE (i <= 11) DO
      WITH TabMois[i] DO
        LeftEdge   := 0;
```

```
      TopEdge     := 0;
      Width       := ImgPtr4^[i].Width;
      Height      := ImgPtr4^[i].Height;
      Depth       := ImgPtr4^[i].Depth;
      ImageData   := ImgPtr4^[i].Data;
      PlanePick   := BYTE (31);
      PlaneOnOff  := BYTE (31);
      NextImage   := NIL;
    END;
    i := i + 1;
  END;
END;
i := 0;
ImgPtr5 := FindImageTable (55555555H,ImgCount5);
IF (ImgPtr5 # NIL) AND (ImgCount5 # 0) THEN
  WHILE (i <= 6) DO
    WITH TabJours[i] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr5^[i].Width;
      Height      := ImgPtr5^[i].Height;
      Depth       := ImgPtr5^[i].Depth;
      ImageData   := ImgPtr5^[i].Data;
      PlanePick   := BYTE (31);
      PlaneOnOff  := BYTE (31);
      NextImage   := NIL;
    END;
    i := i + 1;
  END;
END;
 DateStamp (DSR);
 X := SHORT (DSR.dsDays);
 Y := X;
 X := X + 1;
 AN := 78;
 TrouveAn := FALSE;
 NBJours [0] := 31;
 NBJours [2] := 31;
 NBJours [3] := 30;
 NBJours [4] := 31;
 NBJours [5] := 30;
 NBJours [6] := 31;
 NBJours [7] := 31;
 NBJours [8] := 30;
 NBJours [9] := 31;
 NBJours [10] := 30;
 NBJours [11] := 31;
 NomMois [0] := "Janvier";
 NomMois [1] := "Février";
 NomMois [2] := "Mars";
 NomMois [3] := "Avril";
 NomMois [4] := "Mai";
 NomMois [5] := "Juin";
 NomMois [6] := "Juillet";
 NomMois [7] := "Aout";
 NomMois [8] := "Septembre";
 NomMois [9] := "Octobre";
 NomMois [10] := "Novembre";
 NomMois [11] := "Décembre";
 WHILE NOT TrouveAn DO
    an := real (AN);
    Quatre := 4.0;
    Re := an/Quatre;
    En := entier (Re);
    Re2 := real (En);
    IF Re=Re2
```

```
   THEN
      IF X>366
      THEN
         X := X-366;
         AN := AN+1;
      ELSE
         NBJours [1] := 29;
         TrouveAn := TRUE;
      END;
   ELSE
      IF X>365
      THEN
         X := X-365;
         AN := AN+1;
      ELSE
         NBJours [1]:=28;
         TrouveAn := TRUE;
      END;
   END;
END;
TrouveMois := FALSE;
MOISPREC := 11;
MOIS := 0;
WHILE NOT TrouveMois DO
   IF  X > NBJours [MOIS]
   THEN X := X - NBJours [MOIS];
        MOISPREC := MOIS;
        MOIS := MOIS +1;
   ELSE TrouveMois := TRUE;
   END;
END;
JJour := X;
BeginGadgetList ();
IF Mouvement = "D"
 THEN
  AddGadgetTextButton (2,10,ADR ("Quel jour avez-vous fait cette dépense?")
 ELSE
  AddGadgetTextButton (2,10,ADR ("Quel jour avez-vous fait cette recette?")
END;
  N := NBJours [MOISPREC];
IndiceBlanc := N + JJour;
ABS := 44;
ORD := 80;
i :=0;
indicejour := Y-JJour-N+1;
jourdesem := ((indicejour-1) MOD 7);
ORD := ORD + (jourdesem * 12);
WHILE i<N DO
  AddGadgetImageButton (ABS,ORD,TabJoursVerts[i]);
  i := i+1;
  jourdesem := jourdesem + 1;
  IF jourdesem = 7
  THEN jourdesem :=0;
       ORD := 80;
       ABS := ABS + 19;
  ELSE ORD := ORD + 12;
  END;
END;
N := NBJours [MOIS];
i :=0;
WHILE i < (JJour-1) DO
  TabAffich [i] := TabJoursVerts [i];
  i := i+1;
END;
TabAffich [JJour-1] := TabJoursBlancs [JJour-1];
i := JJour;
```

```
WHILE i < N DO
  TabAffich [i] := TabJoursRouges [i];
  i := i+1;
END;
ABS := 205;
ORD := 80;
i := 0;
ORD := ORD + (jourdesem * 12);
WHILE i < N DO
  AddGadgetImageButton (ABS,ORD,TabAffich[i]);
  i := i + 1;
  jourdesem := jourdesem + 1;
  IF jourdesem = 7
  THEN jourdesem :=0;
       ORD := 80;
       ABS := ABS + 19;
  ELSE ORD := ORD + 12;
  END;
END;

Confirmation := "C'est bien le ";
Gl := EndGadgetList();
IF (Gl # NIL) THEN
     Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,Gl,Scr,NIL);
     DrawImage (Win^.RPort^,TabMois[MOISPREC],64,50);
     DrawImage (Win^.RPort^,TabMois[MOIS],220,50);
     i :=0;
     ORD :=80;
     WHILE i<=6 DO
       DrawImage (Win^.RPort^,TabJours[i],5,ORD);
       DrawImage (Win^.RPort^,TabJours[i],166,ORD);
       i := i+1;
       ORD := ORD + 12;
     END;

   IF (Win # NIL)  THEN
      Rp := Win^.RPort;
      IF Mouvement = "D"
      THEN SetAPen (Rp^,2);
      ELSE SetAPen (Rp^,8);
      END;
      Move (Rp^,0,0);
      Draw (Rp^,0,255);
      Draw (Rp^,319,255);
      Draw (Rp^,319,0);
      Draw (Rp^,0,0);
      SetAPen (Rp^,1);
      IF (TableauParametres[1] = 2) THEN
        IF Mouvement = "D"
         THEN
           Res := SayAndReturn(ADR("kel joor ahvay voo fay set daypos ?"));
         ELSE
           Res := SayAndReturn(ADR("kel joor ahvay voo fay set ra set ?"));
        END;
      END;
      CloseSpeech ();
      Done := FALSE;
      WHILE (NOT Done) DO
        Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
        LOOP
          Msg := GetMsg(Win^.UserPort^);
          IF (Msg = NIL) THEN EXIT; END;
          ProcIMsg (Wp, Msg);
        END;
      END;
      ValeurJour := Trace2;
```

```
          CloseWindow(Win^);
          END;
          FreeGadgetList (G1^);
      CloseScreen(Scr^);
    END;
END ;
END Mois;

(* VAR m: Str20;
    v: CHAR;

BEGIN
    v:= "R";
    Mois (m,v);    *)

END ModMois.
```

```
IMPLEMENTATION MODULE ModMenu;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM MathLib0 IMPORT entier,real;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40, TabNombre,TableauTraces,
        TRecettes,TDepenses,TabNombreInit,TableauParametres,Max,Str100,
        TableauChaine,TableauTemps,AncMinute,AncTick;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                            ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre,Fini,DoneFin,Res : BOOLEAN;
  Win,Won,Win2,Wun : WindowPtr;
  Scr : ScreenPtr;
  Recette,Depense,EtatPM,FIN,Correction : Image;
  TabImages: ARRAY [0..49] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  G1,G1OK,G1Fin : GadgetPtr;
  Wp,WpOK,WpFin : WindowProc;
  Req,ReqOK,ReqFin : Requester;
  Sig : SignalSet;
  Msg,MsgOK,MsgFin : IntuiMessagePtr;
  Somme, SommePrecedente, SommeMax : REAL;
  StrSomme,ES,fonc : Str20;
```

```
  PhrasePron : Str20;
  Phrase : Str100;
  ActMinute,ActTick,ActTick2,ent : INTEGER;
  ree : REAL;
  DSR : DateStampRecord;

PROCEDURE GadgetHandlerFin (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN

  CASE Gad.GadgetID OF
     0 : Fini := TRUE ; DoneFin := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab[TableauChaine.Indice] := 31;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           END;
           Trace := "Il confirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
     1 : Fini := FALSE; DoneFin := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab[TableauChaine.Indice] := 7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           END;
           Trace := "Il infirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
     2 :
  END;
END GadgetHandlerFin;

PROCEDURE GadgetHandlerME (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR P : StringInfoPtr;
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
BEGIN

 IF OpenSpeech() THEN
  IF (Gad.GadgetID = 0)
      THEN fonc := "recettes"; Done := TRUE
```

```
ELSE IF (Gad.GadgetID = 1)
        THEN fonc :=  "depenses"; Done := TRUE
        ELSE IF (Gad.GadgetID = 2)
             THEN fonc := "portemonnaie"; Done := TRUE
             ELSE IF (Gad.GadgetID = 3)
                  THEN fonc := "correction"; Done := TRUE
                  ELSE IF TableauChaine.Indice <= 499
                       THEN TableauChaine.Tab[TableauChaine.Indice] := 30;
                   DateStamp(DSR);
                   ActMinute := SHORT(DSR.dsMinute);
                   ActTick := SHORT(DSR.dsTick);
                   ActTick2 := ActTick;
                   IF ActMinute > AncMinute
                   THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                   END;
                   ent := ActTick - AncTick;
                   ree := real(ent);
                   TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                   AncMinute := ActMinute;
                   AncTick := ActTick2;
                              TableauChaine.Indice := TableauChaine.Indice
                                                      + 1;
                         END;
   Trace := "Il appuie sur FIN ";
     IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
     END;
   BeginGadgetList ();
   GadgetTypeReq := TRUE;
   GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                    RelVerify});
   AddGadgetTextButton (10,40, ADR("OUI"));
   AddGadgetTextButton (135,40, ADR("NON"));
   AddGadgetTextButton (20,10, ADR("AVEZ-VOUS FINI ?"));
   GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
              GadgetMutualExcludeSet {});
   G1Fin := EndGadgetList ();
   InitRequester (ReqFin);
   WITH ReqFin DO
          OlderRequest := NIL;
          LeftEdge := 70;
          TopEdge := 30;
          Width := 180;
          Height := 60;
          ReqGadget := G1Fin;
          ReqBorder := NIL;
          ReqText := NIL;
          Flags := RequesterFlagsSet {};
          BackFill := BYTE(10);
          ReqLayer := NIL;
          ImageBMap := NIL;
   END;
   IF Request (ReqFin, Win^) THEN
      IF (TableauParametres[1] = 2) THEN
          Res := SayAndReturn (ADR("ahvey voo feeny ?"));
      END;
      DoneFin := FALSE;
      WHILE NOT DoneFin DO
          Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
          LOOP
             MsgFin := GetMsg (Win^.UserPort^);
             IF (MsgFin = NIL) THEN EXIT; END;
             ProcIMsg (WpFin, MsgFin);
          END;
      END;
```

```
          END;
          FreeGadgetList (GlFin^);
          Delay(25);
          IF Fini THEN Done := TRUE; fonc := "fin" END;
                    END;
            END;
    END;
  END;
CloseSpeech();
END;
END GadgetHandlerME;


(*****************************************************************************)
(*                                  Menu                                     *)
(*****************************************************************************)

PROCEDURE Menu (VAR FonctionChoisie : Str20);


BEGIN


    WITH Wp DO
      procGadgetUp := GadgetHandlerME;
    END;
    WITH WpFin DO
      procGadgetUp := GadgetHandlerFin;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

      ImgPtr := FindImageTable(22222222H, ImgCount);
```

```
IF (ImgPtr # NIL) AND (ImgCount # 0) THEN

  WITH Recette DO
    LeftEdge    := 0;
    TopEdge     := 0;
    Width       := ImgPtr^[0].Width;
    Height      := ImgPtr^[0].Height;
    Depth       := ImgPtr^[0].Depth;
    ImageData   := ImgPtr^[0].Data;
    PlanePick   := BYTE(31);
    PlaneOnOff  := BYTE(31);
    NextImage := NIL;
  END;
  WITH Depense DO
    LeftEdge    := 0;
    TopEdge     := 0;
    Width       := ImgPtr^[1].Width;
    Height      := ImgPtr^[1].Height;
    Depth       := ImgPtr^[1].Depth;
    ImageData   := ImgPtr^[1].Data;
    PlanePick   := BYTE(31);
    PlaneOnOff  := BYTE(31);
    NextImage := NIL;
  END;
  WITH EtatPM DO
    LeftEdge    := 0;
    TopEdge     := 0;
    Width       := ImgPtr^[2].Width;
    Height      := ImgPtr^[2].Height;
    Depth       := ImgPtr^[2].Depth;
    ImageData   := ImgPtr^[2].Data;
    PlanePick   := BYTE(31);
    PlaneOnOff  := BYTE(31);
    NextImage := NIL;
  END;
  WITH FIN DO
    LeftEdge    := 0;
    TopEdge     := 0;
    Width       := ImgPtr^[3].Width;
    Height      := ImgPtr^[3].Height;
    Depth       := ImgPtr^[3].Depth;
    ImageData   := ImgPtr^[3].Data;
    PlanePick   := BYTE(31);
    PlaneOnOff  := BYTE(31);
    NextImage := NIL;
  END;
  WITH Correction DO
    LeftEdge    := 0;
    TopEdge     := 0;
    Width       := ImgPtr^[4].Width;
    Height      := ImgPtr^[4].Height;
    Depth       := ImgPtr^[4].Depth;
    ImageData   := ImgPtr^[4].Data;
    PlanePick   := BYTE(31);
    PlaneOnOff  := BYTE(31);
    NextImage := NIL;
  END;
  BeginGadgetList();
  AddGadgetImageButton (15,30,Recette);
  AddGadgetImageButton (116,30,Depense);
  AddGadgetImageButton (217,30,EtatPM);
  AddGadgetImageButton (29,130,Correction);
  AddGadgetImageButton (203,130,FIN);

  G1 := EndGadgetList();
  IF (G1 # NIL) THEN
```

```
            Win := CreateWindow (0,0,320,230,WIDCMP,WFlags,Gl,Scr,NIL);
            Won := CreateWindow (20,8,280,15,WIDCMP,WFlags2,NIL,Scr,NIL);
            IF (Win # NIL) AND (Won # NIL) THEN
             IF CreateConsole (Won^) THEN
             wClrScr (Won^);
             Phrase := "Choisissez une activité";
             wSetCursor (Won^,FALSE);
             wMove (Won^,5,1);PutStr (Won^,ADR(Phrase));
             DeleteConsole (Won^);
             END;

             Done := FALSE;
             WHILE (NOT Done) DO
               IF TableauParametres[1] = 2 THEN
                IF OpenSpeech () THEN
                   Res := SayAndReturn (ADR("shwahzee say oon acteeveetay"));
                   CloseSpeech ();
                END;
               END;
               Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
               LOOP
                 Msg := GetMsg(Win^.UserPort^);
                 IF (Msg = NIL) THEN EXIT; END;
                 ProcIMsg (Wp, Msg);
               END;
             END;
           FonctionChoisie := fonc;
           CloseWindow (Won^);
           CloseWindow (Win^);

           END;
           FreeGadgetList (Gl^);
         END;
       CloseScreen(Scr^);
     END;
  END;
END Menu;

END ModMenu.
```

```
IMPLEMENTATION MODULE Scalcu;

FROM SYSTEM IMPORT SHORT;BYTE, ADR;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton,AddGadgetImageButton,GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
       StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3,Str9,Str20,Str40, TabNombre,
   TableauTraces,TRecettes,TDepenses,TabNombreInit,Max,Sactuel,Sinit,Str100,
   TableauParametres,TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Utilitaires IMPORT ConvPrononcable;
FROM RealInOut IMPORT WriteReal;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre,DoneRec,Recom : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff,QR : BOOLEAN
  ActivPoint : BOOLEAN;
(*  For : RealToStringFormat;   *)
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Win3,Win4,Wun : WindowPtr;
```

```
  TacImages: ARRAY [0..58] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3,ImgPtr4,ImgPtr5 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,ImgCount4,ImgCount5,
                                        Indice,list6,list7, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK,GlRec : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK,WpRec : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK,ReqRec : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK,MsgRec : IntuiMessagePtr;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005,
  IRecettes,IDepenses,Fait,I : INTEGER;
  Somme, SommePrecedente, SommeMax ,SA,SAP: REAL;
  ES,fonc,Maxaf : Str20;
  Res,Quit : BOOLEAN;
  DerPron,PPron : Str100;
  StrSomme : Str20;
  CompChiffre : INTEGER;
  OKS : BOOLEAN;
  Snouve,Snouvede,Zero : REAL;
  ActMinute,ActTick,ActTick2,ent : INTEGER;
  ree : REAL;
  DSR : DateStampRecord;
  Rp : RastPortPtr;

PROCEDURE Reel (VAR St:ARRAY OF CHAR; VAR OK :BOOLEAN);

VAR Indice : CARDINAL;
    Premier: BOOLEAN;

BEGIN
    Indice :=0;
    OK :=TRUE;
    Premier :=TRUE;
    IF (CompareString(St,"") = equal ) THEN OK:=FALSE END;
    IF (CompareString(St,"-")= equal ) THEN OK:=FALSE END;
    WHILE ((Indice<StringLength(St)) AND OK ) DO
     IF (( St[Indice]<"0") OR (St[Indice]>"9")) THEN
         IF (((St[Indice]=",")OR(St[Indice]=".")) AND (Premier)) THEN
             St[Indice]:=".";
             Indice :=Indice+1;
             Premier:=FALSE;
         ELSE
           IF ((St[Indice]="-") AND (Indice=StringLength(St)-1)) THEN
             St[Indice]:=" ";
             Indice := Indice+1;
           ELSE OK := FALSE
           END;
         END;
     ELSE
         Indice := Indice+1
     END;
    END;
END Reel;

PROCEDURE GadgetHandlerQuiCA (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadge
BEGIN
  CASE Gad.GadgetID OF
     0 : Quitter := TRUE ; DoneQui := TRUE;
         IF TableauChaine.Indice <= 499
         THEN TableauChaine.Tab [TableauChaine.Indice] := 7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
```

```
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
            Trace := "Il confirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
      1 : Quitter := FALSE; DoneQui := TRUE;
           IF TableauChaine.Indice <= 499
           THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                          = equal)
                 THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
                     ActTick := SHORT(DSR.dsTick);
                     ActTick2 := ActTick;
                     IF ActMinute > AncMinute
                     THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                     END;
                     ent := ActTick - AncTick;
                     ree := real(ent);
                     TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                     AncMinute := ActMinute;
                     AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                 ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
                     ActTick := SHORT(DSR.dsTick);
                     ActTick2 := ActTick;
                     IF ActMinute > AncMinute
                     THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                     END;
                     ent := ActTick - AncTick;
                     ree := real(ent);
                     TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                     AncMinute := ActMinute;
                     AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                 END;
            END;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
      2 :
    END;
END GadgetHandlerQuiCA;

PROCEDURE GadgetHandlerFinCA (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadge
BEGIN
    CASE Gad.GadgetID OF
      0 : Fini := TRUE ; DoneFin := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab [TableauChaine.Indice] := 7;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
```

```
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
        END;
        Trace := "Il confirme la demande ";
        IF TableauTraces.Indice <= 499 THEN
             CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
             TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END; |

1 : Fini := FALSE; DoneFin := TRUE;
     IF TableauChaine.Indice <= 499
     THEN IF NOT QR
          THEN TableauChaine.Tab [TableauChaine.Indice] := 3;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
          ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                     = equal)
                THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
```

```
                END;
            END;
             Trace := "Il infirme la demande ";
             Somme := 0.;
             IF TableauTraces.Indice <= 499 THEN
                  CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                  TableauTraces.Indice :=  TableauTraces.Indice + 1;
             END; |
      2 :
   END;
END GadgetHandlerFinCA;

PROCEDURE GadgetHandlerRecCA (VAR w:Window; VAR Msg :MsgData; VAR Gad: Gadget);
BEGIN
   CASE Gad.GadgetID OF
      0 : Recom := TRUE ; DoneRec := TRUE;
          IF TableauChaine.Indice <= 499
          THEN IF NOT QR
               THEN TableauChaine.Tab [TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
               ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                         = equal)
                    THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    END;
               END;
          END;
           Trace := "Il confirme la demande ";
```

```
        CopyString (StrSomme,  );
        ActivPoint := FALSE;
        IF TableauTraces.Indice <= 499 THEN
             CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
             TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END; |
1 : Recom := FALSE; DoneRec := TRUE;
     IF TableauChaine.Indice <= 499
     THEN IF NOT QR
          THEN TableauChaine.Tab [TableauChaine.Indice] := 3;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                    = equal)
               THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
```

```
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                         TableauChaine.Indice := TableauChaine.Indice + 1;
                    END;
               END;
          END;
           Trace := "Il infirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END; |
     2 :
  END;
END GadgetHandlerRecCA;

PROCEDURE GadgetHandlerCA (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
    SomAct : REAL;
BEGIN
  IF OpenSpeech () THEN
  CASE Gad.GadgetID OF
     0 : ActivEff := TRUE;

         IF (CompChiffre <=8) THEN
          IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
         CompChiffre := CompChiffre + 1;
         IF (TableauParametres[1] = 2) THEN
            Res := SayAndReturn (ADR("zero"));
            DerPron := "zero";
         END;
         wMove (Won^,1,1);
         wClrEndLine (Won^);
         ConcatString (StrSomme,"0");
         wMove (Won^,2,1);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,12,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);
         Trace := "Il appuie sur 0 ";
         IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END;
         END;
         END |
```

```
1 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("an"));
       DerPron := "an";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"1");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 1 ";
    IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END ;

2 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("duh"));
       DerPron := "duh";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"2");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 2 Frs ";
    IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END ;

3 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("trwa"));
       DerPron := "trwa";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"3");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
```

```
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
            Trace := "Il appuie sur 3 ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END |

4 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("katr"));
       DerPron := "katr";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"4");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 4 ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END |

5 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
       Res := SayAndReturn (ADR("senk"));
       DerPron := "senk";
    END;
    wMove (Won^,1,1);
    wClrEndLine (Won^);
    ConcatString (StrSomme,"5");
    wMove (Won^,2,1);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,12,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 5 ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    END;
    END |

6 : ActivEff := TRUE;

    IF (CompChiffre <=8) THEN
     IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
    CompChiffre := CompChiffre + 1;
    IF (TableauParametres[1] = 2) THEN
```

```
      Res := SayAndReturn (ADR("sys"));
      DerPron := "sys";
   END;
   wMove (Won^,1,1);
   wClrEndLine (Won^);
   ConcatString (StrSomme,"6");
   wMove (Won^,2,1);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,12,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);
   Trace := "Il appuie sur 6 ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END;
   END;
   END |

7 : ActivEff := TRUE;

   IF (CompChiffre <=8) THEN
    IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
   CompChiffre := CompChiffre + 1;
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("set"));
      DerPron := "set";
   END;
   wMove (Won^,1,1);
   wClrEndLine (Won^);
   ConcatString (StrSomme,"7");
   wMove (Won^,2,1);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,12,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);
   Trace := "Il appuie sur 7 ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END;
   END;
   END |

8 : ActivEff := TRUE;

   IF (CompChiffre <=8) THEN
    IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
   CompChiffre := CompChiffre + 1;
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("weet"));
      DerPron := "weet";
   END;
   wMove (Won^,1,1);
   wClrEndLine (Won^);
   ConcatString (StrSomme,"8");
   wMove (Won^,2,1);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,12,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);
   Trace := "Il appuie sur 8 ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END;
```

```
      END;
      END ;

 9 : ActivEff := TRUE;

      IF (CompChiffre <=8) THEN
       IF (CompChiffre <=5) OR (ActivPoint = TRUE) THEN
      CompChiffre := CompChiffre + 1;
      IF (TableauParametres[1] = 2) THEN
         Res := SayAndReturn (ADR("nuf"));
         DerPron := "nuf";
      END;
      wMove (Won^,1,1);
      wClrEndLine (Won^);
      ConcatString (StrSomme,"9");
      wMove (Won^,2,1);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,12,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 9 ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      END;
      END ;

10 : Trace := "Il appuie sur Point ";

      IF (CompChiffre <=9) THEN
      CompChiffre := CompChiffre + 1;
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
      END;
      IF ActivPoint = FALSE
      THEN
         IF (TableauParametres[1] = 2 ) THEN
          Res := SayAndReturn (ADR("point"));
          DerPron := "point";
         END;
         wMove (Won^,1,1);
         wClrEndLine (Won^);
         ConcatString (StrSomme,".");
         wMove (Won^,2,1);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,12,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);
         ActivPoint := TRUE;
      END;
      END ;

11 : Trace := "Il appuie sur Virgule ";

      IF (CompChiffre <=9) THEN
      CompChiffre := CompChiffre + 1;
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
      END;
      IF ActivPoint = FALSE
      THEN
         IF (TableauParametres[1] = 2 ) THEN
          Res := SayAndReturn (ADR("virgule"));
          DerPron := "virgule";
         END;
```

```
        wMove (Won^,1,1);
        wClrEndLine (Won^);
        ConcatString (StrSomme,",");
        wMove (Won^,2,1);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,12,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        ActivPoint := TRUE;
    END;
    END ;

12 : Trace := "Il appuie sur OK  ";
     IF TableauChaine.Indice <= 499
     THEN IF NOT QR
          THEN TableauChaine.Tab [TableauChaine.Indice] := 6;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                  = equal)
               THEN TableauChaine.Tab [TableauChaine.Indice] := 23;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
               ELSE TableauChaine.Tab [TableauChaine.Indice] := 15;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
               END;
          END;
     END;
 Reel (StrSomme,OKS);
    IF OKS THEN Somme := ConvStringToReal(StrSomme); END;
    Snouvede := Sactuel-Somme;
    Snouve := Sactuel+Somme;
```

```
      IF TableauTraces.Indice <= 439 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      BeginGadgetList ();
      GadgetTypeReq := TRUE;
      GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                       RelVerify});
      AddGadgetTextButton (10,55, ADR("OUI"));
      AddGadgetTextButton (140,55, ADR("NON"));
      AddGadgetTextButton (10,20, ADR("EST-CE BIEN JUSTE ?"));
      GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                 GadgetMutualExcludeSet {});
      GlFin := EndGadgetList ();
      InitRequester (ReqFin);
      WITH ReqFin DO
            OlderRequest := NIL;
            LeftEdge := 55;
            TopEdge := 88;
            Width := 180;
            Height := 80;
            ReqGadget := GlFin;
            ReqBorder := NIL;
            ReqText := NIL;
            Flags := RequesterFlagsSet {};
            BackFill := BYTE(10);
            ReqLayer := NIL;
            ImageBMap := NIL;
      END;
      IF Request (ReqFin, Win^) THEN
         IF (TableauParametres[1] = 2) THEN
            Res := SayAndReturn (ADR("avay voo feeny ?"));
         END;
         DoneFin := FALSE;
         WHILE NOT DoneFin DO
             Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
             LOOP
                MsgFin := GetMsg (Win^.UserPort^);
                IF (MsgFin = NIL) THEN EXIT; END;
                ProcIMsg (WpFin, MsgFin);
             END;
         END;
      END;

      FreeGadgetList (GlFin^);
      Delay(25);
      IF Fini THEN Done := TRUE END;

13 :   Trace := "Il appuie sur RECOMMENCER ";
       IF TableauChaine.Indice <= 499
       THEN IF NOT QR
            THEN TableauChaine.Tab [TableauChaine.Indice] := 5;
                 DateStamp(DSR);
                 ActMinute := SHORT(DSR.dsMinute);
                 ActTick := SHORT(DSR.dsTick);
                 ActTick2 := ActTick;
                 IF ActMinute > AncMinute
                 THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                 END;
                 ent := ActTick - AncTick;
                 ree := real(ent);
                 TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                 AncMinute := ActMinute;
                 AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
            ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
```

```
                    = equal)
               THEN TableauChaine.Tab [TableauChaine.Indice] := 21;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
               ELSE TableauChaine.Tab [TableauChaine.Indice] := 13;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
               END;
          END;
     END;
     IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
     END;
     BeginGadgetList ();
     GadgetTypeReq := TRUE;
     GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                    RelVerify});
     AddGadgetTextButton (10,55, ADR("OUI"));
     AddGadgetTextButton (180,55, ADR("NON"));
     AddGadgetTextButton (4,20, ADR("VOULEZ-VOUS RECOMMENCER ?"));
     GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
               GadgetMutualExcludeSet {});
     G1Rec := EndGadgetList ();
     InitRequester (ReqRec);
     WITH ReqRec DO
          OlderRequest := NIL;
          LeftEdge := 35;
          TopEdge := 88;
          Width := 220;
          Height := 80;
          ReqGadget := G1Rec;
          ReqBorder := NIL;
          ReqText := NIL;
          Flags := RequesterFlagsSet {};
          BackFill := BYTE(10);
          ReqLayer := NIL;
          ImageBMap := NIL;
     END;
     IF Request (ReqRec, Win^) THEN
        IF (TableauParametres[1] = 2) THEN
          Res := SayAndReturn (ADR("Voolay voo ra comohsay ?"));
        END;
        DoneRec := FALSE;
```

```
        WHILE NOT DoneRec DO
          Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
          LOOP
             MsgRec := GetMsg (Win^.UserPort^);
             IF (MsgRec = NIL) THEN EXIT; END;
             ProcIMsg (WpRec, MsgRec);
          END;
       END;
      END;
      FreeGadgetList (GlRec^);
      Delay(25);
      IF Recom THEN
           CompChiffre :=0;
           wMove (Won^,1,1);
           wClrEndLine (Won^);
           wMove (Won^,2,1);
           PutStr (Won^,ADR(StrSomme));
           wMove (Won^,12,1);
           PutStr (Won^,ADR("Frs"));
           wSetCursor (Won^,FALSE);
           ES := "";
           Der.Last := 0.;
           Der.X := 150;
           AuMoinsUn := FALSE;
           ActivEff := FALSE;
    (*       For := Decimal;    *)
           CopyString (DerPron,PPron);
       END ;

14 : IF QR = TRUE
     THEN
     Trace := "Il appuie sur QUITTER ";
      IF TableauChaine.Indice <= 499
      THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                  = equal)
           THEN TableauChaine.Tab [TableauChaine.Indice] := 22;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           ELSE TableauChaine.Tab [TableauChaine.Indice] := 14;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           END;
      END;
     IF TableauTraces.Indice <= 499 THEN
```

```
            CopyString (TableauTraces.Tab[TableauTraces.Indice],Trace);
            TableauTraces.Indice := TableauTraces.Indice + 1;
        END;
        BeginGadgetList ();
        GadgetTypeReq := TRUE;
        GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                         RelVerify});
        AddGadgetTextButton (10,55, ADR("OUI"));
        AddGadgetTextButton (140,55, ADR("NON"));
        AddGadgetTextButton (2,20, ADR("VOULEZ-VOUS QUITTER ?"));
        GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                   GadgetMutualExcludeSet {});
        GlQui := EndGadgetList ();
        InitRequester (ReqQui);
        WITH ReqQui DO
              OlderRequest := NIL;
              LeftEdge := 55;
              TopEdge := 88;
              Width := 180;
              Height := 80;
              ReqGadget := GlQui;
              ReqBorder := NIL;
              ReqText := NIL;
              Flags := RequesterFlagsSet {};
              BackFill := BYTE(10);
              ReqLayer := NIL;
              ImageBMap := NIL;
        END;
        IF Request (ReqQui, Win^) THEN
           IF (TableauParametres[1] = 2) THEN
              Res := SayAndReturn (ADR("voolay voo kittay ?"));
           END;
           DoneQui := FALSE;
           WHILE NOT DoneQui DO
              Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
              LOOP
                 MsgQui := GetMsg (Win^.UserPort^);
                 IF (MsgQui = NIL) THEN EXIT; END;
                 ProcIMsg (WpQui, MsgQui);
              END;
           END;
        END;
        FreeGadgetList (GlQui^);
        Delay(25);
        IF Quitter THEN Done := TRUE;ES := "quitter" END
      ELSE
           IF (TableauParametres[1] = 2) THEN
              Res := SayAndReturn (ADR(DerPron));
           END;
      END ;

    15 :  IF (TableauParametres[1] = 2) THEN
              Res := SayAndReturn (ADR(DerPron));
           END;
  END;
CloseSpeech ();
END;
END GadgetHandlerCA;

(*****************************************************************************)
(*                          SaisieSommeCalculette                            *)
(*****************************************************************************)

PROCEDURE SaisieSommeCalcu (VAR Phrase : ARRAY OF CHAR;
                            VAR PhrasePron : ARRAY OF CHAR;
                            Quit : BOOLEAN; VAR Valeur : REAL;
```

```
BEGIN
    CopyString (PPron,PhrasePron);
    ES := "";
    QR := Quit;
    Somme := 0.;
    SommeMax := Max;
    Zero := 0.;
    Der.Last := 0.;
    Der.X := 150;
    AuMoinsUn := FALSE;
    ActivEff := FALSE;
    ActivPoint := FALSE;
(*    For := Decimal;      *)
    CopyString (StrSomme,"");
    CompChiffre := 0;

    WITH Wp DO
      procGadgetUp := GadgetHandlerCA;
    END;
    WITH WpQui DO
      procGadgetUp := GadgetHandlerQuiCA;
    END;
    WITH WpFin DO
      procGadgetUp := GadgetHandlerFinCA;
    END;
    WITH WpRec DO
      procGadgetUp := GadgetHandlerRecCA;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 0FCAH;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);
```

```
 ImgPtr := FindImageTable(44333333H, ImgCount);
 IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
   Indice := 0;
   WHILE (Indice <= ImgCount-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr^[Indice].Width;
       Height      := ImgPtr^[Indice].Height;
       Depth       := ImgPtr^[Indice].Depth;
       ImageData   := ImgPtr^[Indice].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     Indice := Indice + 1;
   END;

ImgPtr2 := FindImageTable(44444444H, ImgCount2);
 IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
  WHILE (Indice-ImgCount <= ImgCount2-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr2^[Indice-ImgCount].Width;
       Height      := ImgPtr2^[Indice-ImgCount].Height;
       Depth       := ImgPtr2^[Indice-ImgCount].Depth;
       ImageData   := ImgPtr2^[Indice-ImgCount].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     Indice := Indice + 1;
   END;
 ImgPtr3 := FindImageTable(44777777H, ImgCount3);
 IF (ImgPtr3 # NIL) AND (ImgCount3 # 0) THEN
  WHILE (Indice-ImgCount-ImgCount2 <= ImgCount3-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr3^[Indice-ImgCount-ImgCount2].Width;
       Height      := ImgPtr3^[Indice-ImgCount-ImgCount2].Height;
       Depth       := ImgPtr3^[Indice-ImgCount-ImgCount2].Depth;
       ImageData   := ImgPtr3^[Indice-ImgCount-ImgCount2].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     Indice := Indice + 1;
   END;
ImgPtr4 := FindImageTable(66666666H, ImgCount4);
 IF (ImgPtr4 # NIL) AND (ImgCount4 # 0) THEN
  list6 := 3;
  WHILE (list6 <= ImgCount4-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr4^[list6].Width;
       Height      := ImgPtr4^[list6].Height;
       Depth       := ImgPtr4^[list6].Depth;
       ImageData   := ImgPtr4^[list6].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
```

```
      list6 := list6 + 1;
      Indice := Indice + 1;
    END;
ImgPtr5 := FindImageTable(11111114H, ImgCount5);
 IF (ImgPtr5 # NIL) AND (ImgCount5 # 0) THEN
  list7 := 3;
  WHILE (list7 <= ImgCount5-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr5^[list7].Width;
       Height      := ImgPtr5^[list7].Height;
       Depth       := ImgPtr5^[list7].Depth;
       ImageData   := ImgPtr5^[list7].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list7 := list7 + 1;
     Indice := Indice + 1;
   END;
    BeginGadgetList();
    AddGadgetImageButton (115,147,TabImages[0]);
    AddGadgetImageButton (115,90,TabImages[1]);
    AddGadgetImageButton (135,90,TabImages[2]);
    AddGadgetImageButton (155,90,TabImages[3]);
    AddGadgetImageButton (115,109,TabImages[4]);
    AddGadgetImageButton (135,109,TabImages[5]);
    AddGadgetImageButton (155,109,TabImages[6]);
    AddGadgetImageButton (115,128,TabImages[7]);
    AddGadgetImageButton (135,128,TabImages[8]);
    AddGadgetImageButton (155,128,TabImages[9]);
    AddGadgetImageButton (135,147,TabImages[10]);
    AddGadgetImageButton (155,147,TabImages[28]);
    AddGadgetImageButton (135,220,TabImages[21]);
    AddGadgetImageButton (65,220,TabImages[48]);
    IF Quit = TRUE THEN AddGadgetImageButton (185,220,TabImages[22]) END;

    IF TableauParametres[1] = 2 THEN
          AddGadgetImageButton (5,210,TabImages[47])
    END;
    Gl := EndGadgetList();
    IF (Gl # NIL) THEN
       Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,Gl,Scr,NIL);
       Rp := Win^.RPort;
       IF Quit = FALSE
       THEN SetAPen (Rp^,5);
       ELSE
         IF (CompareString (PPron,"Cobya ah vay voo daypensay ?") = equa
         THEN SetAPen (Rp^,2);
         ELSE SetAPen (Rp^,8);
         END;
       END;
       Move (Rp^,0,0);
       Draw (Rp^,0,255);
       Draw (Rp^,319,255);
       Draw (Rp^,319,0);
       Draw (Rp^,0,0);
       SetAPen (Rp^,1);
       Won := CreateWindow (85,70,130,12,WIDCMP,WFlags,NIL,Scr,NIL);
       Wun := CreateWindow (50,15,264,12,WIDCMP,WFlags2,NIL,Scr,NIL);
       IF (Win # NIL) AND (Won # NIL)  THEN

        IF CreateConsole (Wun^) THEN
          wClrScr (Wun^);
          wSetCursor (Wun^,FALSE);
```

```
              wMove (Wun^,1,1);
              PutStr (Wun^,ADR(Phrase));
              DeleteConsole (Wun^);
            END;

            IF CreateConsole (Won^) THEN

              wClrScr (Won^);
              wMove (Won^,12,1);
              PutStr (Won^,ADR("Frs"));
              wSetCursor (Won^,FALSE);

              IF NOT QR
              THEN
                  DrawImage (Win^.RPort^,TabImages[54],0,0);
              END;
              IF (CompareString (PhrasePron,"Cobya ah vay voo ra su darjo ?")
                      = equal)
              THEN
                DrawImage (Win^.RPort^,TabImages[45],0,0);
              ELSIF (CompareString (PhrasePron,"Cobya ah vay voo daypensay ?"
                      = equal)
              THEN
                DrawImage (Win^.RPort^,TabImages[46],0,0);
              END;
              IF TableauParametres[1] = 2 THEN
                IF OpenSpeech () THEN
                        Res := SayAndReturn (ADR(PhrasePron));
                        CopyString (DerPron,PhrasePron);
                        CloseSpeech ()
                END;
              END;
              Done := FALSE;
              WHILE (NOT Done) DO
                Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                LOOP
                  Msg := GetMsg(Win^.UserPort^);
                  IF (Msg = NIL) THEN EXIT; END;
                  ProcIMsg (Wp, Msg);
                END;
              END;
              DeleteConsole (Won^);
            END;
          EtatSortie := ES;
          Valeur := Somme;
          CloseWindow(Wun^);
          CloseWindow(Won^);
          CloseWindow(Win^);
          END;
          FreeGadgetList (Gl^);
        END;
       CloseScreen(Scr^);
      END;
     END;
    END;
   END;
  END;
END;

END SaisieSommeCalcu;

(*  VAR p: ARRAY [0..40] OF CHAR;
    pp : ARRAY [0..40] OF CHAR;
    q : BOOLEAN;
    v : REAL;
    e : Str20;
```

```
BEGIN
    p :="Combien avez-vous d'argent ?";
    pp := "Cobya ah vay voo darjo ?";
    q := FALSE;
    SaisieSommeCalcu (p,pp,q,v,e);     *)

END Scalcu.
```

```
IMPLEMENTATION MODULE ModMoment;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM MathLib0 IMPORT entier,real;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40, TabNombre,TableauTraces,
        TRecettes,TDepenses,TabNombreInit,TableauParametres,Max,Str100,
        TableauChaine,AncMinute,AncTick,TableauTemps ;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  Scr : ScreenPtr;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Wun : WindowPtr;
  TabImages: ARRAY [0..49] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
```

```
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
  StrSomme,ES,fonc : Str20;
  Res : BOOLEAN;
  PhrasePron : Str100;
  Mo : CHAR;
  DSR : DateStampRecord;
  ent,ActMinute,ActTick,ActTick2 : INTEGER;
  ree : REAL;
  Rp : RastPortPtr;

PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
    0 : OK := TRUE ; DoneOK := TRUE;
        IF TableauChaine.Indice <= 499
        THEN IF Mo = "D"
             THEN TableauChaine.Tab[TableauChaine.Indice] :=19;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
             ELSE TableauChaine.Tab[TableauChaine.Indice] :=11;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
             END;
        END;
        Trace := "Il confirme ";
        IF TableauTraces.Indice <= 499 THEN
             CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
             TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END; |
    1 : OK := FALSE; DoneOK := TRUE;
        IF TableauChaine.Indice <= 499
        THEN IF Mo = "D"
             THEN TableauChaine.Tab[TableauChaine.Indice] :=19;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
```

```
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                     TableauChaine.Tab[TableauChaine.Indice] :=18;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE TableauChaine.Tab[TableauChaine.Indice] :=11;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                     TableauChaine.Tab[TableauChaine.Indice] :=10;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
            END;
            Trace := "Il infirme ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
    2 :
  END;
END GadgetHandlerOK;

PROCEDURE OKRequesterMO (VAR M : Str20;VAR MP : Str20);
BEGIN
            IF OpenSpeech( ) THEN
            Confirmation := "C'est bien l";
            ConcatString (Confirmation,M);
            ConcatString (Confirmation," ?");
            BeginGadgetList ();
            GadgetTypeReq := TRUE;
            GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                             RelVerify});
            AddGadgetTextButton (10,45, ADR("OUI"));
```

```
            AddGadgetTextButton (201,45, ADR("NON"));
            AddGadgetTextButton (15,15, ADR(Confirmation));
            GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
            GlOK := EndGadgetList ();
            InitRequester (ReqOK);
            WITH ReqOK DO
                   OlderRequest := NIL;
                   LeftEdge := 40;
                   TopEdge := 150;
                   Width := 240;
                   Height := 60;
                   ReqGadget := GlOK;
                   ReqBorder := NIL;
                   ReqText := NIL;
                   Flags := RequesterFlagsSet {};
                   BackFill := BYTE(10);
                   ReqLayer := NIL;
                   ImageBMap := NIL;
            END;
            IF Request (ReqOK, Win^) THEN
               IF (TableauParametres[1] = 2) THEN
                 CopyString (PhrasePron,"say bien 1");
                 ConcatString (PhrasePron,MP);
                 Res := SayAndReturn (ADR(PhrasePron));
               END;
               DoneOK := FALSE;
               WHILE NOT DoneOK DO
                    Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                    LOOP
                       MsgOK := GetMsg (Win^.UserPort^);
                       IF (MsgOK = NIL) THEN EXIT; END;
                       ProcIMsg (WpOK, MsgOK);
                    END;
               END;
            END;
            FreeGadgetList (GlOK^);
CloseSpeech();
END;
END OKRequesterMO;


PROCEDURE GadgetHandlerMO (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

BEGIN
  CASE Gad.GadgetID OF
        0 : moment := "e matin";
            momentpron := "a mah tin";
           Trace := "Il appuie sur matin ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterMO (moment,momentpron);
            IF OK THEN Done := TRUE; moment := "matin" END |
        1 : moment := "'après-midi";
            momentpron := "apray meedy";
           Trace := "Il appuie sur apres-midi ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterMO (moment,momentpron);
            IF OK THEN Done := TRUE; moment := "après-midi" END |
        2 : moment := "e soir";
            momentpron := "a swar";
```

```
            Trace := "Il appuie sur soir ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END;
             OKRequesterMO (moment,momentpron);
             IF OK THEN Done := TRUE; moment := "soir" END ;
        3 :
  END;
END GadgetHandlerMO;


(*****************************************************************************)
(*                                  Moment                                   *)
(*****************************************************************************)

PROCEDURE Moment (VAR ValeurMoment : Str20; Mouvement : CHAR);

  BEGIN

    Mo := Mouvement;
    IF OpenSpeech () THEN END;
    WITH Wp DO
      procGadgetUp := GadgetHandlerMO;
    END;
    WITH WpOK DO
      procGadgetUp := GadgetHandlerOK;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);
      ImgPtr := FindImageTable(66666666H, ImgCount);
      IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
        Indice := 0;
```

```
WHILE (Indice <= 2) DO
  WITH TabImages[Indice] DO
    LeftEdge    := 0;
    TopEdge     := 0;
    Width       := ImgPtr^[Indice].Width;
    Height      := ImgPtr^[Indice].Height;
    Depth       := ImgPtr^[Indice].Depth;
    ImageData   := ImgPtr^[Indice].Data;
    PlanePick   := BYTE(31);
    PlaneOnOff  := BYTE(31);
    NextImage := NIL;
  END;
  Indice := Indice + 1;
END;
 Confirmation := "C'est bien 1 ";
 BeginGadgetList();
 AddGadgetImageButton (20,80,TabImages[0]);
 AddGadgetImageButton (114,80,TabImages[1]);
 AddGadgetImageButton (210,80,TabImages[2]);
 IF Mouvement = "D"
     THEN
      AddGadgetTextButton (15,20,
                  ADR("Quand avez-vous fait cette dépense ?"));
     ELSE
      AddGadgetTextButton (15,20,
                  ADR("Quand avez-vous fait cette recette ?"));
 END;
 G1 := EndGadgetList();
 IF (G1 # NIL) THEN
    Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,G1,Scr,NIL);
    IF (Win # NIL)  THEN
     Rp := Win^.RPort;
     IF Mouvement = "D"
     THEN
      SetAPen (Rp^,2);
     ELSE
      SetAPen (Rp^,8);
     END;
     Move (Rp^,0,0);
     Draw (Rp^,0,255);
     Draw (Rp^,319,255);
     Draw (Rp^,319,0);
     Draw (Rp^,0,0);
     SetAPen (Rp^,1);
     IF (TableauParametres[1] = 2) THEN
       IF Mouvement = "D"
          THEN
             Res := SayAndReturn(ADR("kan avay voo fay set daypos ?"))
          ELSE
             Res := SayAndReturn(ADR("kan avay voo fay set ra set ?"));
       END;
     END;
      CloseSpeech ();
     Done := FALSE;
     WHILE (NOT Done) DO
       Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
       LOOP
         Msg := GetMsg(Win^.UserPort^);
         IF (Msg = NIL) THEN EXIT; END;
         ProcIMsg (Wp, Msg);
       END;
     END;
     ValeurMoment := moment;
   CloseWindow(Win^);
   END;
   FreeGadgetList (G1^);
```

```
          END;
        CloseScreen(Scr^);
      END;
  END;
END Moment;

(* VAR
   v : Str20;
   m : CHAR;

BEGIN
 m:= "D";
 Moment (v,m); *)

END ModMoment.
```

```
IMPLEMENTATION MODULE ModEquilibre;

FROM MathLib0 IMPORT entier,real;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, CloseWindow, CloseScreen, ActivateWindow,Image,DrawImage,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, GadgetTypeReq, GlobalGadgetOpt,GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet,Wait;
FROM InOut IMPORT WriteString;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM Strings IMPORT  CopyString, Relation,ConcatString,StringLength,
        CompareString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100, TabNombre,
        TableauTraces,TRecettes,TDepenses,TabNombreInit,TableauParametres,
        Sinit,Sactuel,Max,TableauChaine,TableauTemps,AncMinute,AncTick;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;


CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};

VAR
  Scr : ScreenPtr;
  Req : Requester;
  Gl : GadgetPtr;
  Wp : WindowProc;
  Sig : SignalSet;
  Msg : IntuiMessagePtr;
  Indice : INTEGER;
  Win,Win2,Won,Won2,Wun,Wun2,Wun3 : WindowPtr;
  Phrase,Diff : Str40;
  Done,Q, Res : BOOLEAN;
  Maxaf : Str20;
  cmap : ARRAY [0..31] OF CARDINAL;
  DSR : DateStampRecord;
  ree : REAL;
  ent,ActMinute,ActTick,ActTick2 : INTEGER;

PROCEDURE GadgetHandler (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

BEGIN
  CASE Gad.GadgetID OF
        0 : |
        1 : |
        2 :
                IF TableauChaine.Indice <= 499
                THEN TableauChaine.Tab [TableauChaine.Indice]:= 7;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
```

```
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                                TableauChaine.Indice :=
                                        TableauChaine.Indice + 1;
                            END;

                        Done := TRUE;
                        Q := FALSE; |
        3 :
                         IF TableauChaine.Indice <= 499
                         THEN TableauChaine.Tab[TableauChaine.Indice]
                                        := 32;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                                TableauChaine.Indice :=
                                        TableauChaine.Indice + 1;
                            END;

                        Done := TRUE;
                        Q := TRUE ; |
 END;

END GadgetHandler;

PROCEDURE Equilibre (VAR Quitter : BOOLEAN);

VAR
  VP : REAL;
BEGIN

    WITH Wp DO
      procGadgetUp := GadgetHandler;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
```

```
cmap[13] := 070FH;
cmap[14] := 0C0EH;
cmap[15] := 0C08H;
cmap[16] := 0620H;
cmap[17] := 0E52H;
cmap[18] := 0A52H;
cmap[19] := 0FCAH;
cmap[20] := 0333H;
cmap[21] := 0444H;
cmap[22] := 0555H;
cmap[23] := 0666H;
cmap[24] := 0777H;
cmap[25] := 0888H;
cmap[26] := 0999H;
cmap[27] := 0AAAH;
cmap[28] := 0CCCH;
cmap[29] := 0DDDH;
cmap[30] := 0EEEH;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                        RelVerify});
AddGadgetTextButton (40,20, ADR("VOULEZ-VOUS QUITTER"));
GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
GadgetMutualExcludeSet {});
AddGadgetTextButton (50,40, ADR("LE PROGRAMME ?"));
GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
GadgetMutualExcludeSet {});
AddGadgetTextButton (210,75, ADR("NON"));
AddGadgetTextButton (10,75, ADR("OUI"));

GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                GadgetMutualExcludeSet {});
G1 := EndGadgetList ();
IF G1 # NIL THEN

   Win := CreateWindow (0,0,320,256,WIDCMP,WFlags,NIL,Scr,NIL);
   IF Win # NIL THEN

     InitRequester (Req);
     WITH Req DO
          OlderRequest := NIL;
          LeftEdge := 35;
          TopEdge := 78;
          Width := 250;
          Height := 100;
          ReqGadget := G1;
          ReqBorder := NIL;
          ReqText := NIL;
          Flags := RequesterFlagsSet {};
          BackFill := BYTE(10);
          ReqLayer := NIL;
          ImageBMap := NIL;
     END;
     IF Request(Req,Win^) THEN
      Done := FALSE;
      WHILE (NOT Done) DO
       Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
       LOOP
         Msg := GetMsg(Win^.UserPort^);
         IF (Msg = NIL) THEN EXIT; END;
         ProcIMsg (Wp, Msg);
       END;
```

```
            END;
          END;
          CloseWindow (Win^);
      END;
      Quitter := Q;
      FreeGadgetList (Gl^);
    END;
CloseScreen (Scr^);
END(*Scr*);
END Equilibre;

(*      VAR q : BOOLEAN;

BEGIN
    Equilibre (q);
    IF (q = TRUE) THEN WriteString ("0");
    ELSE WriteString ("1");
    END;          *)
END ModEquilibre.
```

```
IMPLEMENTATION MODULE ModNomSou;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM MathLib0 IMPORT entier,real;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr,ScreenToFront;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenInputFile,
                  CloseInput,Done;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM Conversions IMPORT ConvNumberToString;
FROM VariablesGlobales IMPORT Nom, Str3, Str9,Str20,Str40, TabNombre,
   TableauTraces,TRecettes,TDepenses,TabNombreInit,TableauParametres,Max,Str1C
   TableauChaine,AncMinute,AncTick,TableauTemps ;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done1,DoneOK,OK,OKNombre,Done2,Done3 : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  For : RealToStringFormat;
  Scr : ScreenPtr;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Wun : WindowPtr;
  TabImages: ARRAY [0..49] OF Image;
  ImgPtr : ImageDescTablePtr;
  ImgCount,Indice, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  FileName : ARRAY [0..40] OF CHAR;
  Ms : MenuPtr;
  G1,G12,G13 : GadgetPtr;
```

```
  Wp,Wp2,Wp3 : WindowProc;
  Req,Req2,Req3 : Requester;
  Sig : SignalSet;
  Msg, Msg2, Msg3 : IntuiMessagePtr;
  Somme, SommePrecedente, SommeMax : REAL;
  StrSomme,ES,fonc : Str20;
  Res,CWB : BOOLEAN;
  PhrasePron : Str20;
  JJourStr,ANStr : Str20;
  DSR : DateStampRecord;
  MOIS,MOISPREC,Y : INTEGER;
  TrouveMois,TrouveAn : BOOLEAN;
  NBJours : ARRAY [0..11] OF CARDINAL;
  NomMois : ARRAY [0..11] OF Str9;
  Re,Re2,an : REAL;
  X : CARDINAL;
  Quatre : REAL;
  En : INTEGER;
  JJour,AN : INTEGER;
  Nbrlettre : INTEGER;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree : REAL;

PROCEDURE EnleveBlancs (VAR S : Str20);
  VAR Indice1,Indice2 : CARDINAL;
      S2 : Str20;
  BEGIN
    Indice1 := 0;
    Indice2 := 0;
    WHILE (Indice1 <= (StringLength (S) - 1)) DO
      IF (S[Indice1] # " ") AND (S[Indice1] # "-") AND (S[Indice1] # "_")
      THEN
        IF (S[Indice1] = "é") OR (S[Indice1] = "è")
        THEN S2[Indice2] := "E";
             Indice2 := Indice2 + 1;
        ELSE
           S2[Indice2] := S[Indice1];
           Indice2 := Indice2 + 1;
        END;
      END (* IF *);
      Indice1 := Indice1 + 1;
    END (* WHILE *);
    S2[Indice2] := S[Indice1];
    CopyString (S,S2);
  END EnleveBlancs;

PROCEDURE GadgetHandlerNO (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF
      0 : |
      1 : Done1 := TRUE; Efface := FALSE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab [TableauChaine.Indice] := 2 ;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
```

```
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            CopyString (Trace,"Il confirme le nom");
            IF TableauTraces.Indice <= 499 THEN
                  CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                  TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END;
        2 : Done1 :=TRUE;
            IF TableauChaine.Indice <= 499
            THEN TableauChaine.Tab [TableauChaine.Indice] := 2 ;
                        DateStamp(DSR);
                        ActMinute := SHORT(DSR.dsMinute);
                        ActTick := SHORT(DSR.dsTick);
                        ActTick2 := ActTick;
                        IF ActMinute > AncMinute
                        THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                        END;
                        ent := ActTick - AncTick;
                        ree := real(ent);
                        TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                        AncMinute := ActMinute;
                        AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
                    TableauChaine.Tab [TableauChaine.Indice] := 0;
                        DateStamp(DSR);
                        ActMinute := SHORT(DSR.dsMinute);
                        ActTick := SHORT(DSR.dsTick);
                        ActTick2 := ActTick;
                        IF ActMinute > AncMinute
                        THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                        END;
                        ent := ActTick - AncTick;
                        ree := real(ent);
                        TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                        AncMinute := ActMinute;
                        AncTick := ActTick2;
                    TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            CopyString (Trace,"Il efface le nom");
            IF TableauTraces.Indice <= 499 THEN
                  CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                  TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END
    END;
END GadgetHandlerNO;

PROCEDURE GadgetHandler2 (VAR w: Window; VAR Msg2 : MsgData; VAR Gad: Gadget)

BEGIN

    CASE Gad.GadgetID OF
        0 : ;
        1 : ;
        2 : Done2 :=TRUE;
            IF TableauChaine.Indice <= 499
            THEN TableauChaine.Tab [TableauChaine.Indice] := 0 ;
                        DateStamp(DSR);
                        ActMinute := SHORT(DSR.dsMinute);
                        ActTick := SHORT(DSR.dsTick);
                        ActTick2 := ActTick;
                        IF ActMinute > AncMinute
                        THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute))
                        END;
                        ent := ActTick - AncTick;
                        ree := real(ent);
```

```
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
    END;
END GadgetHandler2;

PROCEDURE GadgetHandlerLettres (VAR w: Window; VAR Msg3 : MsgData;
                                VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF

      0 : Nbrlettre := Nbrlettre + 1;
          IF Nbrlettre <= 20 THEN
             ConcatString (Nom,"A");
             wMove (Won^,1,4);
             wClrEndLine (Won^);
             wMove (Won^,1,4);
             PutStr (Won^,ADR(Nom));
          END; |

      1 : Nbrlettre := Nbrlettre + 1;
          IF Nbrlettre <= 20 THEN
             ConcatString (Nom,"Z");
             wMove (Won^,1,4);
             wClrEndLine (Won^);
             wMove (Won^,1,4);
             PutStr (Won^,ADR(Nom));
          END; |

      2 : Nbrlettre := Nbrlettre + 1;
          IF Nbrlettre <= 20 THEN
             ConcatString (Nom,"E");
             wMove (Won^,1,4);
             wClrEndLine (Won^);
             wMove (Won^,1,4);
             PutStr (Won^,ADR(Nom));
          END; |

      3 : Nbrlettre := Nbrlettre + 1;
          IF Nbrlettre <= 20 THEN
             ConcatString (Nom,"R");
             wMove (Won^,1,4);
             wClrEndLine (Won^);
             wMove (Won^,1,4);
             PutStr (Won^,ADR(Nom));
          END; |

      4 : Nbrlettre := Nbrlettre + 1;
          IF Nbrlettre <= 20 THEN
             ConcatString (Nom,"T");
             wMove (Won^,1,4);
             wClrEndLine (Won^);
             wMove (Won^,1,4);
             PutStr (Won^,ADR(Nom));
          END; |

      5 : Nbrlettre := Nbrlettre + 1;
          IF Nbrlettre <= 20 THEN
             ConcatString (Nom,"Y");
             wMove (Won^,1,4);
             wClrEndLine (Won^);
             wMove (Won^,1,4);
```

```
            PutStr (Won^,ADR(Nom));
        END; |
    6 : Nbrlettre := Nbrlettre + 1;
        IF Nbrlettre <= 20 THEN
            ConcatString (Nom,"U");
            wMove (Won^,1,4);
            wClrEndLine (Won^);
            wMove (Won^,1,4);
            PutStr (Won^,ADR(Nom));
        END; |
    7 : Nbrlettre := Nbrlettre + 1;
        IF Nbrlettre <= 20 THEN
            ConcatString (Nom,"I");
            wMove (Won^,1,4);
            wClrEndLine (Won^);
            wMove (Won^,1,4);
            PutStr (Won^,ADR(Nom));
        END; |
    8 : Nbrlettre := Nbrlettre + 1;
        IF Nbrlettre <= 20 THEN
            ConcatString (Nom,"O");
            wMove (Won^,1,4);
            wClrEndLine (Won^);
            wMove (Won^,1,4);
            PutStr (Won^,ADR(Nom));
        END; |
    9 : Nbrlettre := Nbrlettre + 1;
        IF Nbrlettre <= 20 THEN
            ConcatString (Nom,"P");
            wMove (Won^,1,4);
            wClrEndLine (Won^);
            wMove (Won^,1,4);
            PutStr (Won^,ADR(Nom));
        END; |
   10 : Nbrlettre := Nbrlettre + 1;
        IF Nbrlettre <= 20 THEN
            ConcatString (Nom,"Q");
            wMove (Won^,1,4);
            wClrEndLine (Won^);
            wMove (Won^,1,4);
            PutStr (Won^,ADR(Nom));
        END; |
   11 : Nbrlettre := Nbrlettre + 1;
        IF Nbrlettre <= 20 THEN
            ConcatString (Nom,"S");
            wMove (Won^,1,4);
            wClrEndLine (Won^);
            wMove (Won^,1,4);
            PutStr (Won^,ADR(Nom));
        END; |
   12 : Nbrlettre := Nbrlettre + 1;
        IF Nbrlettre <= 20 THEN
            ConcatString (Nom,"D");
            wMove (Won^,1,4);
            wClrEndLine (Won^);
            wMove (Won^,1,4);
            PutStr (Won^,ADR(Nom));
        END; |
   13 : Nbrlettre := Nbrlettre + 1;
        IF Nbrlettre <= 20 THEN
            ConcatString (Nom,"F");
            wMove (Won^,1,4);
            wClrEndLine (Won^);
            wMove (Won^,1,4);
            PutStr (Won^,ADR(Nom));
        END; |
```

```
14 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
       ConcatString (Nom,"G");
       wMove (Won^,1,4);
       wClrEndLine (Won^);
       wMove (Won^,1,4);
       PutStr (Won^,ADR(Nom));
    END; |
15 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
       ConcatString (Nom,"H");
       wMove (Won^,1,4);
       wClrEndLine (Won^);
       wMove (Won^,1,4);
       PutStr (Won^,ADR(Nom));
    END; |
16 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
       ConcatString (Nom,"J");
       wMove (Won^,1,4);
       wClrEndLine (Won^);
       wMove (Won^,1,4);
       PutStr (Won^,ADR(Nom));
    END; |
17 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
       ConcatString (Nom,"K");
       wMove (Won^,1,4);
       wClrEndLine (Won^);
       wMove (Won^,1,4);
       PutStr (Won^,ADR(Nom));
    END; |
18 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
       ConcatString (Nom,"L");
       wMove (Won^,1,4);
       wClrEndLine (Won^);
       wMove (Won^,1,4);
       PutStr (Won^,ADR(Nom));
    END; |
19 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
       ConcatString (Nom,"M");
       wMove (Won^,1,4);
       wClrEndLine (Won^);
       wMove (Won^,1,4);
       PutStr (Won^,ADR(Nom));
    END; |
20 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
       ConcatString (Nom,"W");
       wMove (Won^,1,4);
       wClrEndLine (Won^);
       wMove (Won^,1,4);
       PutStr (Won^,ADR(Nom));
    END; |
21 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
       ConcatString (Nom,"X");
       wMove (Won^,1,4);
       wClrEndLine (Won^);
       wMove (Won^,1,4);
       PutStr (Won^,ADR(Nom));
    END; |
22 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
```

```
        ConcatString (Nom,"C");
        wMove (Won^,1,4);
        wClrEndLine (Won^);
        wMove (Won^,1,4);
        PutStr (Won^,ADR(Nom));
    END; |
23 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
        ConcatString (Nom,"V");
        wMove (Won^,1,4);
        wClrEndLine (Won^);
        wMove (Won^,1,4);
        PutStr (Won^,ADR(Nom));
    END; |
24 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
        ConcatString (Nom,"B");
        wMove (Won^,1,4);
        wClrEndLine (Won^);
        wMove (Won^,1,4);
        PutStr (Won^,ADR(Nom));
    END; |
25 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
        ConcatString (Nom,"N");
        wMove (Won^,1,4);
        wClrEndLine (Won^);
        wMove (Won^,1,4);
        PutStr (Won^,ADR(Nom));
    END; |
26 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
        ConcatString (Nom,"-");
        wMove (Won^,1,4);
        wClrEndLine (Won^);
        wMove (Won^,1,4);
        PutStr (Won^,ADR(Nom));
    END; |

27 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
        ConcatString (Nom," ");
        wMove (Won^,1,4);
        wClrEndLine (Won^);
        wMove (Won^,1,4);
        PutStr (Won^,ADR(Nom));
    END; |
28 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
        ConcatString (Nom,"E");
        wMove (Won^,1,4);
        wClrEndLine (Won^);
        wMove (Won^,1,4);
        PutStr (Won^,ADR(Nom));
    END; |
29 : Nbrlettre := Nbrlettre + 1;
    IF Nbrlettre <= 20 THEN
        ConcatString (Nom,"E");
        wMove (Won^,1,4);
        wClrEndLine (Won^);
        wMove (Won^,1,4);
        PutStr (Won^,ADR(Nom));
    END; |
30 : Done3 := TRUE;
     Nbrlettre :=0; |
```

```
      31 : Nbrlettre :=0;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab [TableauChaine.Indice] := 1 ;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
                     ActTick := SHORT(DSR.dsTick);
                     ActTick2 := ActTick;
                     IF ActMinute > AncMinute
                     THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                     END;
                     ent := ActTick - AncTick;
                     ree := real(ent);
                     TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                     AncMinute := ActMinute;
                     AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
                TableauChaine.Tab [TableauChaine.Indice] := 0 ;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
                     ActTick := SHORT(DSR.dsTick);
                     ActTick2 := ActTick;
                     IF ActMinute > AncMinute
                     THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                     END;
                     ent := ActTick - AncTick;
                     ree := real(ent);
                     TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                     AncMinute := ActMinute;
                     AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           END;
            CopyString (Nom,"");
           wMove (Won^,1,4);
           wClrEndLine (Won^);

  END;

END GadgetHandlerLettres;

(*****************************************************************************
(*                                Saisie Nom par Souris                      *
(*****************************************************************************

PROCEDURE SaisieNomSou;


TYPE
  TabPtr = POINTER TO ARRAY [0..7] OF INTEGER;

BEGIN
(*     IF OpenSpeech () THEN *)

     TableauChaine.Indice := 0;
     TableauChaine.Tab [TableauChaine.Indice] := 0 ;
     TableauTemps [TableauChaine.Indice] := 0;
     DateStamp(DSR);
     AncMinute := SHORT(DSR.dsMinute);
     AncTick := SHORT(DSR.dsTick);
     TableauChaine.Indice := TableauChaine.Indice + 1;

     WITH Wp DO
      procGadgetUp := GadgetHandlerNO;
    END;
     WITH Wp2 DO
      procGadgetUp := GadgetHandler2;
    END;
```

```
  WITH Wp3 DO
    procGadgetUp := GadgetHandlerLettres;
  END;

  Scr := CreateScreen (320,256,5,NIL);
  IF (Scr # NIL) THEN
    cmap[00] := 0000H;
    cmap[01] := 0FFFH;
    cmap[02] := 0E00H;
    cmap[03] := 0A00H;
    cmap[04] := 0D80H;
    cmap[05] := 0FE0H;
    cmap[06] := 08F0H;
    cmap[07] := 0080H;
    cmap[08] := 00B6H;
    cmap[09] := 00DDH;
    cmap[10] := 00AFH;
    cmap[11] := 007CH;
    cmap[12] := 000FH;
    cmap[13] := 070FH;
    cmap[14] := 0C0EH;
    cmap[15] := 0C08H;
    cmap[16] := 0620H;
    cmap[17] := 0E52H;
    cmap[18] := 0A52H;
    cmap[19] := 0FCAH;
    cmap[20] := 0333H;
    cmap[21] := 0444H;
    cmap[22] := 0555H;
    cmap[23] := 0666H;
    cmap[24] := 0777H;
    cmap[25] := 0888H;
    cmap[26] := 0999H;
    cmap[27] := 0AAAH;
    cmap[28] := 0CCCH;
    cmap[29] := 0DDDH;
    cmap[30] := 0EEEH;
    cmap[31] := 0FFFH;
    LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);


DateStamp (DSR);
X := SHORT (DSR.dsDays);
Y := X;
X := X + 1;
AN := 78;
TrouveAn := FALSE;
NBJours [0] := 31;
NBJours [2] := 31;
NBJours [3] := 30;
NBJours [4] := 31;
NBJours [5] := 30;
NBJours [6] := 31;
NBJours [7] := 31;
NBJours [8] := 30;
NBJours [9] := 31;
NBJours [10] := 30;
NBJours [11] := 31;
NomMois [0] := "Janvier";
NomMois [1] := "Février";
NomMois [2] := "Mars";
NomMois [3] := "Avril";
NomMois [4] := "Mai";
NomMois [5] := "Juin";
NomMois [6] := "Juillet";
NomMois [7] := "Aout";
```

```
NomMois [8] := "Septembre";
NomMois [9] := "Octobre";
NomMois [10] := "Novembre";
NomMois [11] := "Décembre";
WHILE NOT TrouveAn DO
   an := real (AN);
   Quatre := 4.0;
   Re := an/Quatre;
   En := entier (Re);
   Re2 := real (En);
   IF Re=Re2
   THEN
      IF X>366
      THEN
         X := X - 366;
         AN := AN + 1;
      ELSE
         NBJours [1] := 29;
         TrouveAn := TRUE;
      END;
   ELSE
      IF X>365
      THEN
         X := X-365;
         AN := AN+1;
      ELSE
         NBJours [1]:=28;
         TrouveAn := TRUE;
      END;
   END;
END;
TrouveMois := FALSE;
MOISPREC := 11;
MOIS := 0;
WHILE NOT TrouveMois DO
   IF  X > NBJours [MOIS]
   THEN X := X - NBJours [MOIS];
        MOISPREC := MOIS;
        MOIS := MOIS +1;
   ELSE TrouveMois := TRUE;
   END;
END;
JJour := X;
CopyString (Trace,"Date de la session : ");
ConvNumberToString (JJourStr,LONGINT(JJour),FALSE,10,2," ");
ConcatString (Trace,JJourStr);
ConcatString (Trace," ");
ConcatString (Trace,NomMois [MOIS]);
ConcatString (Trace," ");
ConvNumberToString (ANStr,LONGINT(AN),FALSE,10,2," ");
ConcatString (Trace,ANStr);
IF (TableauTraces.Indice <= 499) THEN
  CopyString (TableauTraces.Tab[TableauTraces.Indice],Trace);
  TableauTraces.Indice := TableauTraces.Indice + 1;
END;
ImgPtr := FindImageTable (33335555H,ImgCount);
IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
  Indice :=0;
  WHILE (Indice <= ImgCount-1) DO
   WITH TabImages [Indice] DO
    LeftEdge   := 0;
    TopEdge    := 0;
    Width      := ImgPtr^[Indice].Width;
    Height     := ImgPtr^[Indice].Height;
    Depth      := ImgPtr^[Indice].Depth;
    ImageData  := ImgPtr^[Indice].Data;
```

```
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage   := NIL;
     END;
     Indice :=Indice+1;
    END;
  END;
    BeginGadgetList ();
    AddGadgetImageButton (29,73,TabImages[0]);
    AddGadgetImageButton (54,73,TabImages[1]);
    AddGadgetImageButton (79,73,TabImages[2]);
    AddGadgetImageButton (104,73,TabImages[3]);
    AddGadgetImageButton (129,73,TabImages[4]);
    AddGadgetImageButton (154,73,TabImages[5]);
    AddGadgetImageButton (179,73,TabImages[6]);
    AddGadgetImageButton (204,73,TabImages[7]);
    AddGadgetImageButton (229,73,TabImages[8]);
    AddGadgetImageButton (254,73,TabImages[9]);
    AddGadgetImageButton (40,101,TabImages[10]);
    AddGadgetImageButton (65,101,TabImages[11]);
    AddGadgetImageButton (90,101,TabImages[12]);
    AddGadgetImageButton (115,101,TabImages[13]);
    AddGadgetImageButton (140,101,TabImages[14]);
    AddGadgetImageButton (165,101,TabImages[15]);
    AddGadgetImageButton (190,101,TabImages[16]);
    AddGadgetImageButton (215,101,TabImages[17]);
    AddGadgetImageButton (240,101,TabImages[18]);
    AddGadgetImageButton (265,101,TabImages[19]);
    AddGadgetImageButton (51,129,TabImages[20]);
    AddGadgetImageButton (76,129,TabImages[21]);
    AddGadgetImageButton (101,129,TabImages[22]);
    AddGadgetImageButton (126,129,TabImages[23]);
    AddGadgetImageButton (151,129,TabImages[24]);
    AddGadgetImageButton (176,129,TabImages[25]);
    AddGadgetImageButton (201,129,TabImages[26]);
    AddGadgetImageButton (60,157,TabImages[27]);
    AddGadgetImageButton (226,129,TabImages[28]);
    AddGadgetImageButton (251,129,TabImages[29]);
    AddGadgetImageButton (165,200,TabImages[30]);
    AddGadgetImageButton (105,200,TabImages[31]);

   G13 := EndGadgetList ();
   IF ( G13 # NIL) THEN
   Win := CreateWindow (0,0,320,256,WIDCMP,WFlags,G13,Scr,NIL);
   Won := CreateWindow (70,20,180,50,WIDCMP,WFlags2,NIL,Scr,NIL);
   IF (Won # NIL) THEN
     Efface := TRUE;
     WHILE Efface = TRUE DO
       ActivateWindow (Won^);
       BeginGadgetList();
       GadgetTypeReq := TRUE;
       GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                        RelVerify});
       AddGadgetTextButton (60,35,ADR("EST-CE BIEN JUSTE ?"));
       GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
                        GadgetMutualExcludeSet {});
       AddGadgetTextButton (90,55,ADR("OUI"));
       AddGadgetTextButton (170,55,ADR("NON"));
       G1 := EndGadgetList();
       InitRequester (Req);
       WITH Req DO
         OlderRequest := NIL;
         LeftEdge := 10;
         TopEdge := 70;
         Width := 300;
         Height := 105;
```

```
      ReqGadget := G1;
      ReqBorder := NIL;
      ReqText := NIL;
      Flags := RequesterFlagsSet {};
      BackFill := BYTE(10);
      ReqLayer := NIL;
      ImageBMap := NIL;
    END;

    BeginGadgetList();
    GadgetTypeReq := TRUE;
    GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                     RelVerify});
    AddGadgetTextButton (30,30,ADR("JE NE TROUVE PAS VOTRE NOM"));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
                     GadgetMutualExcludeSet {});
    AddGadgetTextButton (80,45,ADR("ESSAYEZ ENCORE"));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
                     GadgetMutualExcludeSet {});
    AddGadgetTextButton (120,65,ADR("OK"));
    G12 := EndGadgetList();
    InitRequester (Req2);
    WITH Req2 DO
      OlderRequest := NIL;
      LeftEdge := 10;
      TopEdge := 70;
      Width := 300;
      Height := 105;
      ReqGadget := G12;
      ReqBorder := NIL;
      ReqText := NIL;
      Flags := RequesterFlagsSet {};
      BackFill := BYTE(3);
      ReqLayer := NIL;
      ImageBMap := NIL;
    END;

       IF CreateConsole (Won^) THEN
        wMove (Won^,8,1);PutStr(Won^,ADR("BONJOUR !"));
        wMove (Won^,2,1);PutStr(Won^,ADR("QUEL EST VOTRE NOM ?"));
        wMove (Won^,1,4);
        IF (TableauParametres[1] = 2) THEN
(*                Res := SayAndReturn (ADR("Bonjoor, keyl hey votr noh ?"));
      *)        END;
        Nbrlettre :=0;
        CopyString (Nom,"");
        Done3 := FALSE;
        WHILE (NOT Done3) DO
          Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
          LOOP
            Msg3 := GetMsg(Win^.UserPort^);
            IF (Msg3 = NIL) THEN EXIT; END;
            ProcIMsg (Wp3, Msg3);
          END;
        END;
        IF CompareString (Nom,"") # equal THEN EnleveBlancs (Nom); END;
        CopyString (Trace,"Il donne le nom : ");
        ConcatString (Trace,Nom);
        IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
        wSetCursor (Won^,FALSE);
        IF Request (Req, Win^) THEN
        Done1 := FALSE;
        WHILE (NOT Done1) DO
```

```
                Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                LOOP
                  Msg := GetMsg(Win^.UserPort^);
                  IF (Msg = NIL) THEN EXIT; END;
                  ProcIMsg (Wp, Msg);
                END;
              END;
              IF Efface = FALSE (* càd il confirme le nom *)
                 THEN FileName := "Param:";
                      ConcatString (FileName,Nom);
                      ConcatString (FileName,".MAX");
                      OpenInputFile (FileName);
                      IF NOT Done THEN
                              Efface := TRUE;
                              ScreenToFront (Scr^);
                              IF Request (Req2, Win^) THEN
                              Done2 := FALSE;
                              WHILE (NOT Done2) DO
                              Sig := Wait(SignalSet
                                 {CARDINAL(Win^.UserPort^.mpSigBit)});
                                LOOP
                                  Msg2 := GetMsg(Win^.UserPort^);
                                  IF (Msg2 = NIL) THEN EXIT; END;
                                  ProcIMsg (Wp2, Msg2);
                                END;
                              END;
                             ELSE WriteString ("requester pas créé");END;
                            ELSE CloseInput;
                        END;
                    END; (* if efface *)
                  ELSE WriteString ("requester pas créé");END;
             DeleteConsole (Won^);
             END;

            FreeGadgetList (Gl^);
            FreeGadgetList (Gl2^);
            END; (* while efface *)
        CloseWindow(Won^);
        END;
        CloseWindow(Win^);
        CloseScreen(Scr^);
     END;
    END;

(* CloseSpeech ();
END; *)
END SaisieNomSou;

(*  BEGIN
    SaisieNomSou;  *)

END ModNomSou.
```

```
IMPLEMENTATION  MODULE ModJour;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM MathLib0 IMPORT entier,real;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100, TabNombre,
        TableauTraces,TRecettes,TDepenses,TabNombreInit,TableauParametres,Max,
        TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Affichage = RECORD
                Im : Image;
                Jour  : INTEGER;
              END;
VAR
  moment,jour,poste,Nombre,momentpron,jourpron,postepron,VJ : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  For : RealToStringFormat;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Wun : WindowPtr;
  TabAffich: ARRAY [0..6] OF Affichage;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice, i,IndiceBlanc : CARDINAL;
```

```
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005,
  IRecettes,IDepenses,Fait,I : INTEGER;
  Somme, SommePrecedente, SommeMax : REAL;
  StrSomme,ES,fonc : Str20;
  TePtr : IntuiTextPtr;
  Res : BOOLEAN;
  PhrasePron : Str20;
  Mo : CHAR;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree : REAL;
  DSR : DateStampRecord;
  Rp : RastPortPtr;

PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : OK := TRUE ; DoneOK := TRUE;
           IF TableauChaine.Indice <= 499
           THEN IF Mo = "D"
                THEN TableauChaine.Tab[TableauChaine.Indice] :=19;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
                     ActTick := SHORT(DSR.dsTick);
                     ActTick2 := ActTick;
                     IF ActMinute > AncMinute
                     THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                     END;
                     ent := ActTick - AncTick;
                     ree := real(ent);
                     TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                     AncMinute := ActMinute;
                     AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE TableauChaine.Tab[TableauChaine.Indice] :=11;
                     DateStamp(DSR);
                     ActMinute := SHORT(DSR.dsMinute);
                     ActTick := SHORT(DSR.dsTick);
                     ActTick2 := ActTick;
                     IF ActMinute > AncMinute
                     THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                     END;
                     ent := ActTick - AncTick;
                     ree := real(ent);
                     TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                     AncMinute := ActMinute;
                     AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
           END;
           Trace := "Il confirme ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
     1 : OK := FALSE; DoneOK := TRUE;
           IF TableauChaine.Indice <= 499
           THEN IF Mo = "D"
```

```
                THEN TableauChaine.Tab[TableauChaine.Indice] :=19;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                     TableauChaine.Tab[TableauChaine.Indice] :=18;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE TableauChaine.Tab[TableauChaine.Indice] :=11;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                     TableauChaine.Tab[TableauChaine.Indice] :=10;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
            END;
            Trace := "Il infirme ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
    2 :
  END;
END GadgetHandlerOK;
```

```
PROCEDURE OKRequesterJO (VAR J : Str20;VAR JP : Str20);
BEGIN
            IF OpenSpeech () THEN
            Confirmation := "C'est bien le ";
            ConcatString (Confirmation,J);
            ConcatString (Confirmation," ?");
            BeginGadgetList ();
            GadgetTypeReq := TRUE;
            GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                             RelVerify});
            AddGadgetTextButton (10,45, ADR("OUI"));
            AddGadgetTextButton (200,45, ADR("NON"));
            AddGadgetTextButton (15,15, ADR(Confirmation));
            GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
            GlOK := EndGadgetList ();
            InitRequester (ReqOK);
            WITH ReqOK DO
                   OlderRequest := NIL;
                   LeftEdge := 40;
                   TopEdge := 150;
                   Width := 240;
                   Height := 80;
                   ReqGadget := GlOK;
                   ReqBorder := NIL;
                   ReqText := NIL;
                   Flags := RequesterFlagsSet {};
                   BackFill := BYTE(10);
                   ReqLayer := NIL;
                   ImageBMap := NIL;
            END;
            IF Request (ReqOK, Win^) THEN
               IF (TableauParametres[1] = 2) THEN
                  CopyString (PhrasePron,"Say Bien la ");
                  ConcatString (PhrasePron,JP);
                  Res := SayAndReturn (ADR(PhrasePron));
               END;
               DoneOK := FALSE;
               WHILE NOT DoneOK DO
                    Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                    LOOP
                       MsgOK := GetMsg (Win^.UserPort^);
                       IF (MsgOK = NIL) THEN EXIT; END;
                       ProcIMsg (WpOK, MsgOK);
                    END;
               END;
            END;
            FreeGadgetList (GlOK^);
CloseSpeech ();
END;
END OKRequesterJO;

PROCEDURE GadgetHandlerJO ( VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadge
VAR P : StringInfoPtr;
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;

BEGIN
  IF (Gad.GadgetID <> 7)
  THEN
    CASE TabAffich[Gad.GadgetID].Jour OF
        0 : IF (Gad.GadgetID <= IndiceBlanc)
            THEN
              jour := "Lundi";
              jourpron := "lundy";
```

```
      Trace := "Il appuie sur lundi ";
      IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      OKRequesterJO (jour,jourpron);
      IF OK THEN Done := TRUE END;
    END |
1 : IF (Gad.GadgetID <= IndiceBlanc)
    THEN
      jour := "Mardi";
      jourpron := "mardy";
      Trace := "Il appuie sur mardi ";
      IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      OKRequesterJO (jour,jourpron);
      IF OK THEN Done := TRUE END;
    END |
2 : IF (Gad.GadgetID <= IndiceBlanc)
    THEN
      jour := "Mercredi";
      jourpron := "merkrdy";
      Trace := "Il appuie sur mercredi ";
      IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      OKRequesterJO (jour,jourpron);
      IF OK THEN Done := TRUE END;
    END |
3 : IF (Gad.GadgetID <= IndiceBlanc)
    THEN
      jour := "Jeudi";
      jourpron := "judy";
      Trace := "Il appuie sur jeudi ";
      IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      OKRequesterJO (jour,jourpron);
      IF OK THEN Done := TRUE END;
    END |
4 : IF (Gad.GadgetID <= IndiceBlanc)
    THEN
      jour := "Vendredi";
      jourpron := "vendrdy";
      Trace := "Il appuie sur vendredi ";
      IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      OKRequesterJO (jour,jourpron);
      IF OK THEN Done := TRUE END;
    END |
5 : IF (Gad.GadgetID <= IndiceBlanc)
    THEN
      jour := "Samedi";
      jourpron := "samdy";
      Trace := "Il appuie sur samedi ";
      IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      OKRequesterJO (jour,jourpron);
```

```
            IF OK THEN Done := TRUE END;
          END ;
        6 : IF (Gad.GadgetID <= IndiceBlanc)
          THEN
            jour := "Dimanche";
            jourpron := "demansh";
            Trace := "Il appuie sur dimanche ";
            IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END;
            OKRequesterJO (jour,jourpron);
            IF OK THEN Done := TRUE END;
          END;
    END;
  END;
END GadgetHandlerJO;

(*****************************************************************************)
(*                                  Jour                                     *)
(*****************************************************************************)


PROCEDURE Jour (JourDebut : INTEGER; VAR ValeurJour : Str20; Mouvement : CHAR);

VAR ImgPtr,ImgPtr2 : ImageDescTablePtr;
    ImgCount,ImgCount2 : CARDINAL;
    i,I,J,k,l,m : INTEGER;
    Indice1,Indice2,LI1,LI2 : LONGINT;
    TabJoursVerts,TabJoursRouges,TabJoursBlancs : ARRAY[0..6] OF Image;
    DSR : DateStampRecord;

BEGIN

   Mo := Mouvement;
   IF OpenSpeech () THEN END;
    WITH Wp DO
      procGadgetUp := GadgetHandlerJO;
    END;
    WITH WpOK DO
      procGadgetUp := GadgetHandlerOK;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
```

```
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

  i := 0;
  ImgPtr := FindImageTable (88888888H,ImgCount);
  IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
    WHILE (i <= 6) DO
      WITH TabJoursVerts[i] DO
        LeftEdge   := 0;
        TopEdge    := 0;
        Width      := ImgPtr^[i].Width;
        Height     := ImgPtr^[i].Height;
        Depth      := ImgPtr^[i].Depth;
        ImageData  := ImgPtr^[i].Data;
        PlanePick  := BYTE (31);
        PlaneOnOff := BYTE (31);
        NextImage  := NIL;
      END;
      i := i + 1;
   END;
   WHILE (i <= 13) DO
      WITH TabJoursRouges[i-7] DO
        LeftEdge   := 0;
        TopEdge    := 0;
        Width      := ImgPtr^[i].Width;
        Height     := ImgPtr^[i].Height;
        Depth      := ImgPtr^[i].Depth;
        ImageData  := ImgPtr^[i].Data;
        PlanePick  := BYTE (31);
        PlaneOnOff := BYTE (31);
        NextImage  := NIL;
      END;
      i := i + 1;
   END;
   i := 0;
   ImgPtr2 := FindImageTable (99999999H,ImgCount2);
   IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
    WHILE (i <= 6) DO
      WITH TabJoursBlancs[i] DO
        LeftEdge   := 0;
        TopEdge    := 0;
        Width      := ImgPtr2^[i].Width;
        Height     := ImgPtr2^[i].Height;
        Depth      := ImgPtr2^[i].Depth;
        ImageData  := ImgPtr2^[i].Data;
        PlanePick  := BYTE (31);
        PlaneOnOff := BYTE (31);
        NextImage  := NIL;
      END;
      i := i + 1;
   END;

   DateStamp (DSR);
   LI1 := 1D;
   LI2 := 7D;
   Indice1 := ((DSR.dsDays - LI1) MOD LI2);
   Indice2 := ((DSR.dsDays - LI1) MOD LI2);
```

```
I := SHORT (Indice1);
I := I - JourDebut;
IF (I < 0) THEN I := 7 + I END;
IndiceBlanc := I;
J := SHORT (Indice2);
TabAffich[I].Im := TabJoursBlancs[J];
TabAffich[I].Jour := J;
I := I - 1;
WHILE (I >= 0) DO
  J := J - 1;
  IF (J < 0) THEN J := 6 END;
  TabAffich[I].Im := TabJoursVerts[J];
  TabAffich[I].Jour := J;
  I := I - 1;
END;
I := SHORT (Indice1) - JourDebut;
IF (I < 0) THEN I := 7 + I END;
I := I + 1;
J := SHORT (Indice2);
WHILE (I <= 6) DO
  J := J + 1;
  IF (J > 6) THEN J := 0 END;
  TabAffich[I].Im := TabJoursRouges[J];
  TabAffich[I].Jour := J;
  I := I + 1;
END;

Confirmation := "C'est bien le ";
BeginGadgetList ();
k := 0;
l:=4;
WHILE (k <= 6) DO
  AddGadgetImageButton (l,100,TabAffich[k].Im);
  l := l + 45;
  k := k + 1;
END;
IF Mouvement = "D"
 THEN
  AddGadgetTextButton (2,20,ADR ("Quel jour avez-vous fait cette dépense?")
 ELSE
  AddGadgetTextButton (2,20,ADR ("Quel jour avez-vous fait cette recette?")
END;
G1 := EndGadgetList();
IF (G1 # NIL) THEN
     Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,G1,Scr,NIL);
     IF (Win # NIL)  THEN
      Rp := Win^.RPort;
      IF Mouvement = "D"
      THEN SetAPen (Rp^,2);
      ELSE SetAPen (Rp^,8);
      END;
      Move (Rp^,0,0);
      Draw (Rp^,0,255);
      Draw (Rp^,319,255);
      Draw (Rp^,319,0);
      Draw (Rp^,0,0);
      SetAPen (Rp^,1);
      IF (TableauParametres[1] = 2) THEN
        IF Mouvement = "D"
         THEN
           Res := SayAndReturn(ADR("kel joor ahvay voo fay set daypos ?"))
         ELSE
           Res := SayAndReturn(ADR("kel joor ahvay voo fay set ra set ?"))
        END;
      END;
      CloseSpeech ();
```

```
          Done := FALSE;
          WHILE (NOT Done) DO
            Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
            LOOP
              Msg := GetMsg(Win^.UserPort^);
              IF (Msg = NIL) THEN EXIT; END;
              ProcIMsg (Wp, Msg);
            END;
          END;
          ValeurJour := jour;
          CloseWindow(Win^);
          END;
          FreeGadgetList (G1^);
        END;
       END;
      END;
      CloseScreen(Scr^);
    END;
END Jour;

(*  VAR
   j : INTEGER;
   v : Str20;
   m : CHAR;
BEGIN
   j := 2;
   m := "D";
   Jour (j,v,m);  *)

END ModJour.
```

```
IMPLEMENTATION MODULE SRNombre;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100, TabNombreInit,
                              TableauTraces,TRecettes,TDepenses,
                              TableauParametres,Sinit,Max,TabNombre,Sactuel;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr,FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                              ConvStringToReal;
FROM Utilitaires IMPORT ConvPrononcable;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};

VAR
  Done,DoneOK : BOOLEAN;
  Gl,GlOK : GadgetPtr;
  Wp,WpOK : WindowProc;
  Req,ReqOK : Requester;
  Sig : SignalSet;
  Msg,MsgOK : IntuiMessagePtr;
  ValeurMax,ValeurPrec : REAL;
  Win,Won,Wun,Win1,Win2,Win3,Win4 : WindowPtr;
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  Res : BOOLEAN;
  SomPron,PhrasePron : Str100;
  Phrase,Phrase3,Maxaf  : Str20;
  Phrase2,Diff : ARRAY [0..39] OF CHAR;
  Rp : RastPortPtr;
  PorteMon : Image;
  ImgPtr : ImageDescTablePtr;
```

```
  ImgCount : CARDINAL;

PROCEDURE GadgetHandler (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN

  Done := TRUE;
END GadgetHandler;

PROCEDURE SommeRestanteNombre (EPMReq : INTEGER);


BEGIN
  IF OpenSpeech () THEN END;
  ValeurMax := Max;
  WITH Wp DO
    procGadgetUp := GadgetHandler;
  END;
  Scr := CreateScreen (320,256,5,NIL);
  IF (Scr # NIL) THEN
          cmap[00] := 0000H;
          cmap[01] := 0FFFH;
          cmap[02] := 0E00H;
          cmap[03] := 0A00H;
          cmap[04] := 0D80H;
          cmap[05] := 0FE0H;
          cmap[06] := 08F0H;
          cmap[07] := 0080H;
          cmap[08] := 00B6H;
          cmap[09] := 00DDH;
          cmap[10] := 00AFH;
          cmap[11] := 007CH;
          cmap[12] := 000FH;
          cmap[13] := 070FH;
          cmap[14] := 0C0EH;
          cmap[15] := 0C08H;
          cmap[16] := 0620H;
          cmap[17] := 0E52H;
          cmap[18] := 0A52H;
          cmap[19] := 0FCAH;
          cmap[20] := 0333H;
          cmap[21] := 0444H;
          cmap[22] := 0555H;
          cmap[23] := 0666H;
          cmap[24] := 0777H;
          cmap[25] := 0888H;
          cmap[26] := 0999H;
          cmap[27] := 0AAAH;
          cmap[28] := 0CCCH;
          cmap[29] := 0DDDH;
          cmap[30] := 0EEEH;
          cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);
      ImgPtr := FindImageTable(11111114H, ImgCount);
      IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
          WITH PorteMon DO
            LeftEdge    := 0;
            TopEdge     := 0;
            Width       := ImgPtr^[6].Width;
            Height      := ImgPtr^[6].Height;
            Depth       := ImgPtr^[6].Depth;
            ImageData   := ImgPtr^[6].Data;
            PlanePick   := BYTE(31);
            PlaneOnOff  := BYTE(31);
            NextImage := NIL;
          END;
      END;
```

```
      BeginGadgetList();
      AddGadgetTextButton (1,1,ADR(" SUITE "));
      Gl := EndGadgetList();
      IF (Gl # NIL) THEN

      Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,NIL,Scr,NIL);
      Won := CreateWindow (93,60,180,100,WIDCMP,WFlags2,NIL,Scr,NIL);
      Wun := CreateWindow (132,230,100,24,WIDCMP,WFlags2,Gl,Scr,NIL);
      IF (Won # NIL) AND (Win # NIL) THEN
         Rp := Win^.RPort;
         SetAPen (Rp^,5);
         Move (Rp^,0,0);
         Draw (Rp^,0,255);
         Draw (Rp^,319,255);
         Draw (Rp^,319,0);
         Draw (Rp^,0,0);
         SetAPen (Rp^,1);
         DrawImage (Win^.RPort^,PorteMon,0,0);
         ActivateWindow (Won^);
         IF CreateConsole (Won^) THEN
             wSetCursor (Won^,FALSE);
             wMove (Won^,2,1);PutStr(Won^,ADR("Il vous reste : "));
             ConvRealToString (Sinit,Phrase,2,Decimal);
             CopyString (Phrase3,Phrase);
             ConcatString (Phrase," Francs");
             wMove (Won^,2,4);PutStr(Won^,ADR(Phrase));
             DeleteConsole (Won^);
         END;
         ValeurPrec := 0.;
         IF (TableauParametres[4] = 4) THEN
          CopyString (PhrasePron,"il voo rest ");
          ConvPrononcable (Phrase3,SomPron);
          ConcatString (PhrasePron,SomPron);
          Res := SayAndReturn (ADR (PhrasePron));
         END;

        CloseSpeech();
      Done := FALSE;
      WHILE (NOT Done) DO
       Sig := Wait(SignalSet{CARDINAL(Wun^.UserPort^.mpSigBit)});
       LOOP
         Msg := GetMsg(Wun^.UserPort^);
         IF (Msg = NIL) THEN EXIT; END;
         ProcIMsg (Wp, Msg);
       END;
      END;

       CloseWindow(Wun^);
       CloseWindow(Won^);
       CloseWindow(Win^);
     END;
     FreeGadgetList (Gl^);
     END;
     CloseScreen(Scr^);
     END;

END SommeRestanteNombre;

(*    VAR
    e: INTEGER;

BEGIN
    e := 1;
    SommeRestanteNombre (e);       *)

END SRNombre.
```

```
IMPLEMENTATION MODULE SRBillets;

FROM MathLib0 IMPORT entier,real;
FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet,GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
       StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40, TabNombreInit,TableauTra
s,
                              TRecettes,TDepenses,TableauParametres,Sinit,Ma
                              TabNombre,Sactuel,Str100;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM Drawing IMPORT SetAPen,Draw,Move;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                            ConvStringToReal;
FROM Utilitaires IMPORT ConvPrononcable;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;

VAR
  Done,DoneOK : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  Gl : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
```

```
  Nb5000,Nb1000,Nb500,Nb100,Nb50,Nb20,Nb5,Nb1 : CARDINAL;
  Confirmation,Trace : Str40;
  Der : Dernier;
  Somme, SommePrecedente, SommeMax,ValeurMax,ValeurPrec : REAL;
  StrSomme,ES,fonc,SommeString,Maxaf : Str20;
  TePtr : IntuiTextPtr;
  For : RealToStringFormat;
  Win,Win1,Win2,Win3 : WindowPtr;
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  Mini : ARRAY [0..8] OF Image;
  Icone : ARRAY [0..3] OF Image;
  ImgPtr,ImgPtr2 : ImageDescTablePtr;
  ImgCount,ImgCount2,Indice,i,I,X,Y : CARDINAL;
  Res : BOOLEAN;
  SomPron,PhrasePron : Str100;
  Phrase  : Str40;
  Phrase2,Diff : ARRAY [0..39] OF CHAR;
  Rp : RastPortPtr;
  ree,ree2,re2 : REAL;
  ent : INTEGER;

PROCEDURE GadgetHandler (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

BEGIN

  Done := TRUE;

END GadgetHandler;

(*****************************************************************************)
(*                              SommeRestanteBillets                         *)
(*****************************************************************************)

PROCEDURE SommeRestanteBillets (EPMReq : INTEGER);

BEGIN

  IF OpenSpeech () THEN END;
  ValeurMax := Max;

  WITH Wp DO
      procGadgetUp := GadgetHandler;
  END;

  Scr := CreateScreen (320,256,5,NIL);
  IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 0FCAH;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0E00H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
```

```
    cmap[19] := 0FCAH;
    cmap[20] := 0333H;
    cmap[21] := 0444H;
    cmap[22] := 0555H;
    cmap[23] := 0666H;
    cmap[24] := 07D8H;
    cmap[25] := 0C97H;
    cmap[26] := 09CFH;
    cmap[27] := 0D9EH;
    cmap[28] := 0EB0H;
    cmap[29] := 0DDDH;
    cmap[30] := 0EC7H;
    cmap[31] := 0FFFH;
    LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);



    ImgPtr := FindImageTable(15111111H, ImgCount);
    IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
      Indice := 0;
      WHILE (Indice <= ImgCount-1) DO
        WITH Mini[Indice] DO
          LeftEdge    := 0;
          TopEdge     := 0;
          Width       := ImgPtr^[Indice].Width;
          Height      := ImgPtr^[Indice].Height;
          Depth       := ImgPtr^[Indice].Depth;
          ImageData   := ImgPtr^[Indice].Data;
          PlanePick   := BYTE(31);
          PlaneOnOff  := BYTE(31);
          NextImage := NIL;
        END;
        Indice := Indice + 1;
      END;


    ImgPtr2 := FindImageTable(14111111H, ImgCount2);
    IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
      Indice := 0;
      WHILE (Indice <= ImgCount2 -1) DO
        WITH Icone[Indice] DO
          LeftEdge    := 0;
          TopEdge     := 0;
          Width       := ImgPtr2^[Indice].Width;
          Height      := ImgPtr2^[Indice].Height;
          Depth       := ImgPtr2^[Indice].Depth;
          ImageData   := ImgPtr2^[Indice].Data;
          PlanePick   := BYTE(31);
          PlaneOnOff  := BYTE(31);
          NextImage := NIL;
        END;
        Indice := Indice + 1;
      END;

  BeginGadgetList();
  AddGadgetTextButton (2,4,ADR ("SUITE"));
  G1 := EndGadgetList ();
  IF (G1 # NIL) THEN
   Win  := CreateWindow (0,0,320,256,WIDCMP,WFlags2,NIL,Scr,NIL);
   IF (Win # NIL) THEN
    Win1 := CreateWindow (75,11,242,15,WIDCMP,WFlags2,NIL,Scr,NIL);
    Win2 := CreateWindow (75,27,242,15,WIDCMP,WFlags2,NIL,Scr,NIL);
    Win3 := CreateWindow (132,230,100,24,WIDCMP,WFlags2,G1,Scr,NIL);
    IF (Win1 # NIL) AND (Win2 # NIL) AND (Win3 # NIL) THEN
```

```
Rp := Win^.RPort;
ConvRealToString (Sinit,SommeString,0,Decimal);
IF (TableauParametres[4] = 3) OR (TableauParametres[4] = 6)
THEN
  IF CreateConsole (Win1^)
  THEN
     CopyString (Phrase,"Vous avez  ");
     ConcatString (Phrase,SommeString);
     ConcatString (Phrase," Francs");
     wSetCursor (Win1^,FALSE);
     wMove (Win1^,1,1);
     PutStr (Win1^,ADR(Phrase));
     DeleteConsole (Win1^);
  END;
  IF CreateConsole (Win2^)
  THEN
     CopyString (Phrase,"dans votre porte-monnaie");
     wSetCursor (Win2^,FALSE);
     wMove (Win2^,1,1);
     PutStr (Win2^,ADR(Phrase));
     DeleteConsole (Win2^);
  END;
END;
DrawImage (Win^.RPort^,Icone [0],0,0);
SetAPen (Rp^,5);
Move (Rp^,0,0);
Draw (Rp^,0,255);
Draw (Rp^,319,255);
Draw (Rp^,319,0);
Draw (Rp^,0,0);
SetAPen (Rp^,1);

IF (TableauParametres[4] = 5) OR (TableauParametres[4] = 6)
THEN
    CopyString (PhrasePron,"voo ah vay ");
    ConvPrononcable (SommeString,SomPron);
    ConcatString (PhrasePron,SomPron);
     Res := SayAndReturn (ADR (PhrasePron));
END;
CloseSpeech();

Nb5000 := 1;
WHILE (Nb5000 <= TabNombreInit [0]) DO

   IF Nb5000 <= 16
   THEN
      ree := real (Nb5000);
      ree2 := ree/2.;
      ent := entier (ree2);
      re2 := real (ent);
      IF ree2 = re2
      THEN
        Der.Y := 50;
        Der.X := 24 + ((Nb5000 - 2) * 18);
        Der.X := Der.X + ((Nb5000 - 2) DIV 2);
      ELSE
        Der.Y := 57;
        Der.X := 5 + ((Nb5000 - 1) * 19);
        Der.X := Der.X - ((Nb5000 - 1) DIV 2);
      END;
      DrawImage (Win^.RPort^,Mini[0],Der.X,Der.Y)
   END;

   Nb5000 := Nb5000 + 1;
   END;
```

```
    Nb1000 :=1;
    WHILE (Nb1000 <= TabNombreInit [1]) DO

    IF Nb1000 <= 17
    THEN
       ree := real (Nb1000);
       ree2 := ree/2.;
       ent := entier (ree2);
       re2 := real (ent);
       IF ree2 = re2
       THEN
         Der.Y := 75;
         Der.X := 23 + ((Nb1000 - 2) * 17);
         Der.X := Der.X + ((Nb1000 - 2) DIV 2);
       ELSE
         Der.Y := 82;
         Der.X := 5 + ((Nb1000 - 1) * 18);
         Der.X := Der.X - ((Nb1000 - 1) DIV 2);
       END;
       DrawImage (Win^.RPort^,Mini[1],Der.X,Der.Y)
    END;
    Nb1000 := Nb1000 +1;
    END;

    Nb500 :=1;
    WHILE (Nb500 <= TabNombreInit [2]) DO

    IF Nb500 <= 18
    THEN
       ree := real (Nb500);
       ree2 := ree/2.;
       ent := entier (ree2);
       re2 := real (ent);
       IF ree2 = re2
       THEN
         Der.Y := 100;
         Der.X := 22 + ((Nb500 - 2) * 16);
         Der.X := Der.X + ((Nb500 - 2) DIV 2);
       ELSE
         Der.Y := 107;
         Der.X := 5 + ((Nb500 - 1) * 17);
         Der.X := Der.X - ((Nb500 - 1) DIV 2);
       END;
       DrawImage (Win^.RPort^,Mini[2],Der.X,Der.Y)
    END;
    Nb500 := Nb500 + 1;
    END;

  Nb100 :=1;
  WHILE (Nb100 <= TabNombreInit [3]) DO

    IF Nb100 <= 19
    THEN
       ree := real (Nb100);
       ree2 := ree/2.;
       ent := entier (ree2);
       re2 := real (ent);
       IF ree2 = re2
       THEN
         Der.Y := 125;
         Der.X := 21 + ((Nb100 - 2) * 15);
         Der.X := Der.X + ((Nb100 - 2) DIV 2);
       ELSE
         Der.Y := 132;
         Der.X := 5 + ((Nb100 - 1) * 16);
         Der.X := Der.X - ((Nb100 - 1) DIV 2);
```

```
            END;
            DrawImage (Win^.RPort^,Mini[3],Der.X,Der.Y)
        END;
        Nb100 := Nb100 + 1;
      END;

      Nb50 :=1;
      WHILE (Nb50 <= TabNombreInit [5]) DO

        IF Nb50 <= 21
        THEN
            Der.X := 5 + ((Nb50 - 1) * 15);
            Der.Y := 150;
            DrawImage (Win^.RPort^,Mini[4],Der.X,Der.Y)
        END;
        Nb50 := Nb50 + 1;
      END;

      Nb20 :=1;
      WHILE (Nb20 <= TabNombreInit [6]) DO
        IF Nb20 <= 17
        THEN
            Der.X := 5 + ((Nb20 - 1) * 18);
            Der.Y := 168;
            DrawImage (Win^.RPort^,Mini [5],Der.X,Der.Y)
        END;
        Nb20 := Nb20 + 1;
      END;

      Nb5 := 1;
      WHILE (Nb5 <= TabNombreInit [7]) DO
        IF Nb5 <= 19
        THEN
            Der.X := 5 + ((Nb5 - 1) * 16);
            Der.Y := 189;
            DrawImage (Win^.RPort^,Mini[6],Der.X,Der.Y)
        END;
        Nb5 := Nb5 + 1;
       END;

       Nb1 :=1;
       WHILE (Nb1 <= TabNombreInit [8]) DO
        IF Nb1 <= 28
        THEN
            Der.X := 5 + ((Nb1 - 1) * 11);
            Der.Y := 208;
            DrawImage (Win^.RPort^,Mini[7],Der.X,Der.Y)
        END;
        Nb1 := Nb1 + 1;
       END;

     Done := FALSE;
     WHILE (NOT Done) DO
       Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
       LOOP
         Msg := GetMsg(Win3^.UserPort^);
         IF (Msg = NIL) THEN EXIT; END;
         ProcIMsg (Wp, Msg);
       END;
     END;
     CloseWindow(Win1^);
     CloseWindow(Win2^);
     CloseWindow(Win3^);
   END;
   CloseWindow(Win^);
END;
```

```
    FreeGadgetList (Gl^);
   END;
  END;
  END;
  CloseScreen(Scr^);
  END;

END SommeRestanteBillets;

(*      VAR u: INTEGER;

BEGIN
   u := 1;
   SommeRestanteBillets (u);    *)

END SRBillets.
```

```
IMPLEMENTATION MODULE ModPoste;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM MathLib0 IMPORT real,entier;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40, TabNombre,TableauTraces,
        TRecettes,TDepenses,TabNombreInit,TableauParametres,Max,Str100,
        TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM Drawing IMPORT SetAPen,Move,Draw;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";

VAR
  moment,jour,poste,Nombre,Affichage,momentpron,jourpron,postepron : Str20;
  Confirmation,Trace : Str40;
  Done,DoneOK,OK,OKNombre : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  For : RealToStringFormat;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Recette,Depense,EtatPM,FIN : Image;
  Win,Won,Win2,Wun : WindowPtr;
  TabImages: ARRAY [0..49] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  Gl,GlEff,GlQui,GlFin,GlOK : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
```

```
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005,
  IRecettes,IDepenses,Fait,I : INTEGER;
  Somme, SommePrecedente, SommeMax : REAL;
  StrSomme,ES,fonc : Str20;
  TePtr : IntuiTextPtr;
  Res : BOOLEAN;
  PhrasePron : Str100;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree : REAL;
  DSR : DateStampRecord;
  Rp : RastPortPtr;

PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : OK := TRUE ; DoneOK := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab[TableauChaine.Indice] :=17;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
           END;
           Trace := "Il confirme ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END; |
    1 : OK := FALSE; DoneOK := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab[TableauChaine.Indice] :=17;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
                TableauChaine.Tab[TableauChaine.Indice] :=16;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
```

```
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il infirme ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |
    2 :
  END;
END GadgetHandlerOK;

PROCEDURE OKRequesterPO (VAR P : Str20;VAR PP : Str20);
BEGIN
            IF OpenSpeech() THEN
            Confirmation := "C'est bien pour ";
            ConcatString (Confirmation,P);
            ConcatString (Confirmation," ?");
            BeginGadgetList ();
            GadgetTypeReq := TRUE;
            GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                             RelVerify});
            AddGadgetTextButton (10,45, ADR("OUI"));
            AddGadgetTextButton (268,45, ADR("NON"));
            AddGadgetTextButton (2,15, ADR(Confirmation));
            GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
            GlOK := EndGadgetList ();
            InitRequester (ReqOK);
            WITH ReqOK DO
                   OlderRequest := NIL;
                   LeftEdge := 6;
                   TopEdge := 160;
                   Width := 308;
                   Height := 60;
                   ReqGadget := GlOK;
                   ReqBorder := NIL;
                   ReqText := NIL;
                   Flags := RequesterFlagsSet {};
                   BackFill := BYTE(10);
                   ReqLayer := NIL;
                   ImageBMap := NIL;
            END;
            IF Request (ReqOK, Win^) THEN
               IF (TableauParametres[1] = 2) THEN
                 CopyString (PhrasePron,"say bien poor ");
                 ConcatString (PhrasePron,PP);
                 Res := SayAndReturn (ADR(PhrasePron));
               END;
               DoneOK := FALSE;
               WHILE NOT DoneOK DO
                   Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                   LOOP
                      MsgOK := GetMsg (Win^.UserPort^);
                      IF (MsgOK = NIL) THEN EXIT; END;
                      ProcIMsg (WpOK, MsgOK);
                   END;
               END;
            END;
            FreeGadgetList (GlOK^);
CloseSpeech();
END;
```

```
END OKRequesterPO;

(*****************************************************************************)
(*                                GadgetHandlers                             *)
(*****************************************************************************)

PROCEDURE GadgetHandlerPO (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR P : StringInfoPtr;
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
BEGIN
  CASE Gad.GadgetID OF
        0 : poste := "de la nourriture";
            postepron := "da lah nooree ter";
           Trace := "Il appuie sur alimentation ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Alimentation" END |
        1 : poste := "des habits";
            postepron := "dayza bee";
           Trace := "Il appuie sur vêtements ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Vêtements" END |
        2 : poste := "des loisirs";
            postepron := "day lwazyr";
           Trace := "Il appuie sur loisirs ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Loisir" END |
        3 : poste := "du tabac";
            postepron := "du tahbah";
           Trace := "Il appuie sur tabac ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Tabac" END |
        4 : poste := "des choses diverses";
            postepron := "day sho zdeevyrs";
           Trace := "Il appuie sur divers ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Divers" END |
         5 : poste := "des transports";
            postepron := "day trohspoar";
           Trace := "Il appuie sur transport ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END;
            OKRequesterPO (poste,postepron);
            IF OK THEN Done := TRUE; poste := "Transport" END |
         6 : poste := "la tirelire";
```

```
            postepron := "lah teer leer";
            Trace := "Il appuie sur tirelire ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END;
             OKRequesterPO (poste,postepron);
             IF OK THEN Done := TRUE; poste := "Tirelire" END ;
        7 : poste := "des magazines";
             postepron := "day mah gah zeen";
            Trace := "Il appuie sur magazines ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END;
             OKRequesterPO (poste,postepron);
             IF OK THEN Done := TRUE; poste := "Magazines" END ;
        8 :
  END;
END GadgetHandlerPO;


(*****************************************************************************)
(*                                  Poste                                    *)
(*****************************************************************************)

PROCEDURE Poste (VAR ValeurPoste : Str20);


BEGIN

    IF OpenSpeech () THEN END;
    WITH Wp DO
      procGadgetUp := GadgetHandlerPO;
    END;
    WITH WpOK DO
      procGadgetUp := GadgetHandlerOK;
    END;
    Scr := CreateScreen (320,256,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
```

```
cmap[26] := 0999H;
cmap[27] := 0AAAH;
cmap[28] := 0CCCH;
cmap[29] := 0DDDH;
cmap[30] := 0EEEH;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);
ImgPtr := FindImageTable(11111111H, ImgCount);
IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
  Indice := 0;
  WHILE (Indice <= 7) DO
    WITH TabImages[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr^[Indice].Width;
      Height      := ImgPtr^[Indice].Height;
      Depth       := ImgPtr^[Indice].Depth;
      ImageData   := ImgPtr^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;
   Confirmation := "C'est bien le ";
   BeginGadgetList();
   AddGadgetImageButton (20,40,TabImages[0]);
   AddGadgetImageButton (88,40,TabImages[1]);
   AddGadgetImageButton (156,40,TabImages[2]);
   AddGadgetImageButton (224,40,TabImages[3]);
   AddGadgetImageButton (20,100,TabImages[4]);
   AddGadgetImageButton (88,100,TabImages[5]);
   AddGadgetImageButton (156,100,TabImages[6]);
   AddGadgetImageButton (224,100,TabImages[7]);
   AddGadgetTextButton (2,20,
                        ADR("Pour quoi avez-vous fait cette dépense?"));
   G1 := EndGadgetList();
   IF (G1 # NIL) THEN
      Win := CreateWindow (0,0,320,256,WIDCMP,WFlags2,G1,Scr,NIL);
      IF (Win # NIL)  THEN
       Rp := Win^.RPort;
       SetAPen (Rp^,2);
       Move (Rp^,0,0);
       Draw (Rp^,0,255);
       Draw (Rp^,319,255);
       Draw (Rp^,319,0);
       Draw (Rp^,0,0);
       SetAPen (Rp^,1);
       IF (TableauParametres[1] = 2) THEN
          Res := SayAndReturn (ADR("Poor kwaha vey voo fay set daypos ?
;
       END;
       CloseSpeech ();
       Done := FALSE;
       WHILE (NOT Done) DO
         Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
         LOOP
           Msg := GetMsg(Win^.UserPort^);
           IF (Msg = NIL) THEN EXIT; END;
           ProcIMsg (Wp, Msg);
         END;
       END;
     ValeurPoste := poste;
     CloseWindow(Win^);
     END;
     FreeGadgetList (G1^);
```

```
          END;
        CloseScreen(Scr^);
      END;
  END;

END Poste;

(* VAR v : Str20;

BEGIN
   Poste (v);  *)

END ModPoste.
```

```
IMPLEMENTATION MODULE Utilitaires;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str100, Str20,Str40,TabNombreInit,
                TableauTraces,TRecettes,TDepenses,TableauParametres,Sinit,Max,
                TabNombre,Sactuel;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                                ConvStringToReal;


CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";

(*************************************************************************)

PROCEDURE ConvPrononcable (VAR Chi : Str20 ; VAR Pro : Str100);

VAR PEntiere, PDecimale,Unite,Dizaine,Centaine,Millier,Decime,Centime : Str20
    P,L : INTEGER;
    C,U,Cent : CHAR;
    SpecialUnite,SpecialCentaine,SpecialMillier,SpecialFranc,SpecialDecimale,
    SpecialCentime,Res : BOOLEAN;

BEGIN

   P := LocateChar (Chi,".",0,19);
   IF P = -1
    THEN CopyString (PEntiere,Chi);PDecimale :=""
    ELSE ExtractSubString (PEntiere,Chi,0,P);
         ExtractSubString (PDecimale,Chi,P+1,19-P);
   END;
```

```
  L := StringLength (PEntiere);
  IF L = 0 THEN SpecialFranc := TRUE
           ELSE SpecialFranc := FALSE
  END;
  IF L-1 >= 0
    THEN
       C := PEntiere [L-1];
       CASE C OF

       "0" : U := "0"; Unite := "";IF L = 1 THEN SpecialFranc := TRUE END |
       "1" : U := "1"; Unite := "an" |
       "2" : U := "2"; Unite := "duh" |
       "3" : U := "3"; Unite := "trwa" |
       "4" : U := "4"; Unite := "katr" |
       "5" : U := "5"; Unite := "senk" |
       "6" : U := "6"; Unite := "sys" |
       "7" : Unite := "set" |
       "8" : Unite := "weet" |
       "9" : Unite := "nuf" |
       ELSE

       END;
  END;
  IF L-2 >= 0
    THEN
       SpecialUnite := FALSE;
       C := PEntiere [L-2];
       CASE C OF

       "0" : Dizaine := "" |
       "1" :   CASE U OF

               "0" : Dizaine := "dhe"; |
               "1" : SpecialUnite := TRUE;
                     Dizaine := "onz" |
               "2" : SpecialUnite := TRUE;
                     Dizaine := "dooz" |
               "3" : SpecialUnite := TRUE;
                     Dizaine := "trayze" |
               "4" : SpecialUnite := TRUE;
                     Dizaine := "katorz" |
               "5" : SpecialUnite := TRUE;
                     Dizaine := "kenz" |
               "6" : SpecialUnite := TRUE;
                     Dizaine := "says" |
               "7" : Dizaine := "dys" |
               "8" : Dizaine := "dys" |
               "9" : Dizaine := "dys" |
               ELSE

               END; |

       "2" : Dizaine := "vant" |
       "3" : Dizaine := "tran't" |
       "4" : Dizaine := "karan't" |
       "5" : Dizaine := "sekan't" |
       "6" : Dizaine := "swah san't" |
       "7" : Dizaine := "septan't" |
       "8" : Dizaine := "katru van" |
       "9" : Dizaine := "nonan't" |
       ELSE

       END;
  END;
  IF L-3 >= 0
    THEN
```

```
      SpecialCentaine := FALSE;
      C := PEntiere [L-3];
      CASE C OF

      "0" : SpecialCentaine := TRUE;
            Centaine := "" |
      "1" : Centaine := "" |
      "2" : Centaine := "duh" |
      "3" : Centaine := "trwa" |
      "4" : Centaine := "katr" |
      "5" : Centaine := "senk" |
      "6" : Centaine := "sys" |
      "7" : Centaine := "set" |
      "8" : Centaine := "weet" |
      "9" : Centaine := "naf" |
      ELSE

      END;
   ELSE SpecialCentaine := TRUE;
END;
IF L-4 >= 0
   THEN
      SpecialMillier := FALSE;
      C := PEntiere [L-4];
      CASE C OF

      "0" : SpecialMillier := TRUE;
            Millier := "" |
      "1" : Millier := "" |
      "2" : Millier := "duh" |
      "3" : Millier := "trwa" |
      "4" : Millier := "katr" |
      "5" : Millier := "senk" |
      "6" : Millier := "sys" |
      "7" : Millier := "set" |
      "8" : Millier := "weet" |
      "9" : Millier := "naf" |
      ELSE

      END;
   ELSE SpecialMillier := TRUE;
END;

L := StringLength (PDecimale);
IF L # 0
   THEN
      SpecialDecimale := FALSE;
      C := PDecimale [1];
      IF C = "0" THEN Cent := "0"; Centime := ""
                 ELSIF C = "5" THEN Cent := "5"; Centime := "senk";
                                    SpecialCentime := FALSE;
                               ELSE Centime := ""
      END;
      C := PDecimale [0];
      CASE C OF

      "0" : Decime := "" |
      "1" : IF Cent = "0" THEN Decime := "dhe"
                          ELSE Decime := "kens"; SpecialCentime := TRUE
            END; |
      "2" : Decime := "vant" |
      "3" : Decime := "tran't" |
      "4" : Decime := "karan't" |
      "5" : Decime := "sekan't" |
      "6" : Decime := "swasan't" |
      "7" : Decime := "septan't" |
```

```
        "8" : Decime := "katru van" |
        "9" : Decime := "nonan't" |
        ELSE
        END;
        IF (CompareString(PDecimale,"0") = equal) OR
                (CompareString(PDecimale,"00") = equal) THEN
                        SpecialDecimale := TRUE
        END;
      ELSE SpecialDecimale := TRUE
    END;

    Pro := "";
    IF NOT SpecialFranc
      THEN
        IF NOT SpecialMillier
                THEN ConcatString (Pro,Millier);
                     ConcatString (Pro,"myl");
        END;
        IF NOT SpecialCentaine
                THEN ConcatString (Pro,Centaine);
                     ConcatString (Pro,"soh");
        END;
        ConcatString (Pro,Dizaine);
        IF NOT SpecialUnite
                THEN ConcatString (Pro,Unite);
        END;
        ConcatString (Pro,"froh'");
    END;
    IF NOT SpecialDecimale
      THEN
        ConcatString (Pro,Decime);
        IF NOT SpecialCentime
                THEN ConcatString (Pro,Centime);
        END;
        ConcatString (Pro,"santeam");
    END;

END ConvPrononcable;

(*************************************************************************)

PROCEDURE RegletteJour (VAR W:Window ; JourDebut:INTEGER ;
                        Horizontal:BOOLEAN ; X,Y:CARDINAL);

VAR ImgPtr,ImgPtr2 : ImageDescTablePtr;
    ImgCount,ImgCount2 : CARDINAL;
    i,I,J : INTEGER;
    Indice1,Indice2,LI1,LI2 : LONGINT;
    TabJoursVerts,TabJoursRouges,TabJoursBlancs,TabAffich : ARRAY[0..6] OF Im
;
    DSR : DateStampRecord;

BEGIN
  i := 0;
  ImgPtr := FindImageTable (88888888H,ImgCount);
  IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
    WHILE (i <= 6) DO
      WITH TabJoursVerts[i] DO
        LeftEdge   := 0;
        TopEdge    := 0;
        Width      := ImgPtr^[i].Width;
        Height     := ImgPtr^[i].Height;
        Depth      := ImgPtr^[i].Depth;
        ImageData  := ImgPtr^[i].Data;
        PlanePick  := BYTE (31);
```

```
      PlaneOnOff := BYTE (31);
      NextImage  := NIL;
    END;
    i := i + 1;
END;
WHILE (i <= 13) DO
    WITH TabJoursRouges[i-7] DO
      LeftEdge   := 0;
      TopEdge    := 0;
      Width      := ImgPtr^[i].Width;
      Height     := ImgPtr^[i].Height;
      Depth      := ImgPtr^[i].Depth;
      ImageData  := ImgPtr^[i].Data;
      PlanePick  := BYTE (31);
      PlaneOnOff := BYTE (31);
      NextImage  := NIL;
    END;
    i := i + 1;
END;
i := 0;
ImgPtr2 := FindImageTable (99999999H,ImgCount2);
IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
 WHILE (i <= 6) DO
    WITH TabJoursBlancs[i] DO
      LeftEdge   := 0;
      TopEdge    := 0;
      Width      := ImgPtr2^[i].Width;
      Height     := ImgPtr2^[i].Height;
      Depth      := ImgPtr2^[i].Depth;
      ImageData  := ImgPtr2^[i].Data;
      PlanePick  := BYTE (31);
      PlaneOnOff := BYTE (31);
      NextImage  := NIL;
    END;
    i := i + 1;
END;

DateStamp (DSR);
LI1 := 1D;
LI2 := 7D;
Indice1 := ((DSR.dsDays - LI1) MOD LI2);
Indice2 := ((DSR.dsDays - LI1) MOD LI2);
I := SHORT (Indice1);
I := I - JourDebut;
IF (I < 0) THEN I := 7 + I END;
J := SHORT (Indice2);
TabAffich[I] := TabJoursBlancs[J];
I := I - 1;
WHILE (I >= 0) DO
  J := J - 1;
  IF (J < 0) THEN J := 6 END;
  TabAffich[I] := TabJoursVerts[J];
  I := I - 1;
END;
I := SHORT (Indice1) - JourDebut;
IF (I < 0) THEN I := 7 + I END;
I := I + 1;
J := SHORT (Indice2);
WHILE (I <= 6) DO
  J := J + 1;
  IF (J > 6) THEN J := 0 END;
  TabAffich[I] := TabJoursRouges[J];
  I := I + 1;
END;

IF Horizontal
```

```
     THEN
       i := 0;
       WHILE (i <= 6) DO
         DrawImage (W.RPort^,TabAffich[i],X,Y);
         X := X + 41;
         i := i + 1;
       END;
     ELSE
       i := 0;
       WHILE (i <= 6) DO
         DrawImage (W.RPort^,TabAffich[i],X,Y);
         Y := Y + 15;
         i := i + 1;
       END;
     END;
   END;
   END;
END RegletteJour;

(*************************************************************************)

PROCEDURE RegHeure (VAR W:Window ; HORIZONTAL:BOOLEAN ; X,Y:CARDINAL);

VAR
   Regl, ImBlanc, ImRouge : Image;
   ImgPtr : ImageDescTablePtr;
   ImgCount : CARDINAL;
   DSR : DateStampRecord;
   NbreMin,JR,JB,Int,i,I : INTEGER;

BEGIN
   ImgPtr := FindImageTable (11111112H,ImgCount);
   IF (ImgPtr # NIL) AND (ImgCount # 0) THEN

     IF HORIZONTAL THEN

        WITH Regl DO
         LeftEdge   := 0;
         TopEdge    := 0;
         Width      := ImgPtr^[0].Width;
         Height     := ImgPtr^[0].Height;
         Depth      := ImgPtr^[0].Depth;
         ImageData  := ImgPtr^[0].Data;
         PlanePick  := BYTE (31);
         PlaneOnOff := BYTE (31);
         NextImage  := NIL;
        END;
        WITH ImBlanc DO
         LeftEdge   := 0;
         TopEdge    := 0;
         Width      := ImgPtr^[1].Width;
         Height     := ImgPtr^[1].Height;
         Depth      := ImgPtr^[1].Depth;
         ImageData  := ImgPtr^[1].Data;
         PlanePick  := BYTE (31);
         PlaneOnOff := BYTE (31);
         NextImage  := NIL;
        END;
        WITH ImRouge DO
         LeftEdge   := 0;
         TopEdge    := 0;
         Width      := ImgPtr^[2].Width;
         Height     := ImgPtr^[2].Height;
         Depth      := ImgPtr^[2].Depth;
         ImageData  := ImgPtr^[2].Data;
         PlanePick  := BYTE (31);
```

```
      PlaneOnOff := BYTE (31);
      NextImage  := NIL;
     END;

                ELSE
     WITH Reg1 DO
      LeftEdge   := 0;
      TopEdge    := 0;
      Width      := ImgPtr^[3].Width;
      Height     := ImgPtr^[3].Height;
      Depth      := ImgPtr^[3].Depth;
      ImageData  := ImgPtr^[3].Data;
      PlanePick  := BYTE (31);
      PlaneOnOff := BYTE (31);
      NextImage  := NIL;
     END;
     WITH ImBlanc DO
      LeftEdge   := 0;
      TopEdge    := 0;
      Width      := ImgPtr^[4].Width;
      Height     := ImgPtr^[4].Height;
      Depth      := ImgPtr^[4].Depth;
      ImageData  := ImgPtr^[4].Data;
      PlanePick  := BYTE (31);
      PlaneOnOff := BYTE (31);
      NextImage  := NIL;
     END;
     WITH ImRouge DO
      LeftEdge   := 0;
      TopEdge    := 0;
      Width      := ImgPtr^[5].Width;
      Height     := ImgPtr^[5].Height;
      Depth      := ImgPtr^[5].Depth;
      ImageData  := ImgPtr^[5].Data;
      PlanePick  := BYTE (31);
      PlaneOnOff := BYTE (31);
      NextImage  := NIL;
     END;
END;

DateStamp (DSR);
NbreMin := SHORT (DSR.dsMinute);
JR := (NbreMin DIV 60) * 4;
Int := NbreMin MOD 60;
    IF (Int >= 0)  AND (Int <= 15) THEN JB := 1
ELSIF (Int > 15)  AND (Int <= 30) THEN JB := 2
ELSIF (Int > 30)  AND (Int <= 45) THEN JB := 3
                                  ELSE JB := 4
END;
IF NOT HORIZONTAL THEN DrawImage (W.RPort^,Reg1,X-1,Y-124);
                  ELSE DrawImage (W.RPort^,Reg1,X-6,Y-1);
END;
I := 1;
WHILE (I < JR) DO
  i := 1;
  WHILE (i <= 4) DO
    DrawImage (W.RPort^,ImRouge,X,Y);
    IF HORIZONTAL THEN X := X + 1
                  ELSE Y := Y - 1
    END;
    i := i + 1;
    I := I + 1;
  END;
  IF HORIZONTAL THEN X := X + 1
                ELSE Y := Y - 1
  END;
```

```
    END;

    I := 1;
    WHILE (I <= JB) DO
      DrawImage (W.RPort^,ImBlanc,X,Y);
      IF HORIZONTAL THEN X := X + 1
                    ELSE Y := Y - 1
      END;
      I := I + 1;
    END;
   END;
END RegHeure;

END Utilitaires.
```

```
IMPLEMENTATION MODULE VariablesGlobales;

BEGIN
END VariablesGlobales.
```

```
DEFINITION MODULE EtatPorteMonnaie;

PROCEDURE EPM;

END EtatPorteMonnaie.


DEFINITION MODULE EnregistrerSomme;

FROM VariablesGlobales IMPORT Str20;

PROCEDURE EnregistrerRecette;
PROCEDURE EnregistrerDepense;
PROCEDURE EnregistrerSommeInitiale ( VAR EtatSortie2 : Str20);

END EnregistrerSomme.


DEFINITION MODULE Interface;

FROM VariablesGlobales IMPORT Str20;

PROCEDURE SSaisieNom;
PROCEDURE Menu (VAR FonctionChoisie : Str20);
PROCEDURE Recettes (VAR ValeurRecette : REAL; VAR EtatSortie1 : Str20);
PROCEDURE Depenses (VAR ValeurDepense : REAL; VAR EtatSortie1 : Str20);
PROCEDURE SommeInitiale (VAR ValeurSommeInitiale : REAL;
                         VAR EtatSortie1 : Str20);
PROCEDURE PorteMonnaie;
PROCEDURE EPMRecette;
PROCEDURE EPMDepense;
PROCEDURE MomentDepense (VAR ValeurMoment : Str20; Mouvement : CHAR);
PROCEDURE PosteDepense (VAR ValeurPoste : Str20);
PROCEDURE PosteRecette (VAR ValeurPoste : Str20);
PROCEDURE PresentationRecDep;
PROCEDURE CorrectionPM;
END Interface.


DEFINITION MODULE ModPresentation;

PROCEDURE Presentation;

END ModPresentation.


DEFINITION MODULE ModReglette;

PROCEDURE Reglette (Orig_Mvt : CHAR; Affectation:BOOLEAN);

END ModReglette.
```

```
DEFINITION MODULE CorrectionPMCA;

FROM VariablesGlobales IMPORT Str20;


PROCEDURE CorrPMCA (VAR Phrase : ARRAY OF CHAR ;
                    VAR PhrasePron : ARRAY OF CHAR );

END CorrectionPMCA.
DEFINITION MODULE CorrPMBillets;

FROM VariablesGlobales IMPORT Str20;


PROCEDURE CorrPMBI (VAR Phrase : ARRAY OF CHAR ;
                                 VAR PhrasePron : ARRAY OF CHAR );


END CorrPMBillets.
DEFINITION MODULE CorrectionPMCL;

FROM VariablesGlobales IMPORT Str20;


PROCEDURE CorrPMCL (VAR Phrase : ARRAY OF CHAR ;
                    VAR PhrasePron : ARRAY OF CHAR );


END CorrectionPMCL.



DEFINITION MODULE SSNombre;

FROM VariablesGlobales IMPORT Str20;


PROCEDURE SaisieSommeNbre (VAR Phrase : ARRAY OF CHAR ;
                                 VAR PhrasePron : ARRAY OF CHAR ;
                                 Quit : BOOLEAN ; VAR Valeur : REAL ;
                                 VAR EtatSortie : Str20);
END SSNombre.



DEFINITION MODULE SSBillets;

FROM VariablesGlobales IMPORT Str20;


PROCEDURE SaisieSommeBillets (VAR Phrase : ARRAY OF CHAR ;
                                 VAR PhrasePron : ARRAY OF CHAR ;
                                 Quit : BOOLEAN ; VAR Valeur : REAL ;
                                 VAR EtatSortie : Str20);
END SSBillets.
```

```
DEFINITION MODULE ModNom;

PROCEDURE SaisieNom;

END ModNom.


DEFINITION MODULE ModPosteRec;

FROM VariablesGlobales IMPORT Str20;

PROCEDURE PosteRec (VAR Poste : Str20);

END ModPosteRec.


DEFINITION MODULE ModMois;

FROM VariablesGlobales IMPORT Str20;

PROCEDURE Mois ( VAR Jour : Str20 ; Mouvement : CHAR);

END ModMois.


DEFINITION MODULE ModMenu;

FROM VariablesGlobales IMPORT Str20;

PROCEDURE Menu (VAR FonctionChoisie : Str20);

END ModMenu.


DEFINITION MODULE Scalcu;

FROM VariablesGlobales IMPORT Str20;

PROCEDURE SaisieSommeCalcu (VAR Phrase : ARRAY OF CHAR ;
                            VAR PhrasePron : ARRAY OF CHAR ;
                            Quit : BOOLEAN ; VAR Valeur : REAL ;
                            VAR EtatSortie : Str20);
END Scalcu.
```

```
DEFINITION MODULE ModMoment;
FROM VariablesGlobales IMPORT Str20;
PROCEDURE Moment (VAR Moment : Str20; Mouvement : CHAR);
END ModMoment.

DEFINITION MODULE ModEquilibre;
PROCEDURE Equilibre (VAR Quitter : BOOLEAN);
END ModEquilibre.

DEFINITION MODULE ModNomSou;
PROCEDURE SaisieNomSou;
END ModNomSou.

DEFINITION MODULE ModJour;
FROM VariablesGlobales IMPORT Str20;
PROCEDURE Jour (JourDebut : INTEGER ; VAR Jour : Str20 ; Mouvement : CHAR);
END ModJour.

DEFINITION MODULE SRNombre;
PROCEDURE SommeRestanteNombre(EPMReq : INTEGER);
END SRNombre.

DEFINITION MODULE SRBillets;
PROCEDURE SommeRestanteBillets(EPMReq : INTEGER);
END SRBillets.

DEFINITION MODULE ModPoste;
FROM VariablesGlobales IMPORT Str20;
PROCEDURE Poste (VAR Poste : Str20);
END ModPoste.
```

```
DEFINITION MODULE Utilitaires;

FROM Intuition IMPORT Window;
FROM VariablesGlobales IMPORT Str20,Str100;


PROCEDURE ConvPrononcable (VAR Chi : Str20; VAR Pro : Str100);
PROCEDURE RegletteJour (VAR W:Window; JourDebut : INTEGER; Horizontal:BOOLEAN;
                        X,Y : CARDINAL);
PROCEDURE RegHeure (VAR W : Window ;HORIZONTAL : BOOLEAN; X,Y : CARDINAL);

END Utilitaires.



DEFINITION MODULE VariablesGlobales;

TYPE Str3 = ARRAY [0..2] OF CHAR;
TYPE Str9 = ARRAY [0..8] OF CHAR;
TYPE Str20 = ARRAY [0..19] OF CHAR;
TYPE Str40 = ARRAY [0..39] OF CHAR;
TYPE Str60 = ARRAY [0..59] OF CHAR;
TYPE Str100 = ARRAY [0..99] OF CHAR;
TYPE DepRec = RECORD
              Valeur : REAL;
              Jour : Str20;
              Poste : Str20;
            END;
VAR TDepenses : RECORD
                  Tab : ARRAY [0..49] OF DepRec;
                  Indice : INTEGER;
                END;
VAR TRecettes :  RECORD
                  Tab : ARRAY [0..49] OF DepRec;
                  Indice : INTEGER;
                 END;
VAR TableauParametres : ARRAY [0..19] OF INTEGER;
VAR TableauTraces : RECORD
                      Tab : ARRAY [0..499] OF Str40;
                      Indice : INTEGER;
                    END;
VAR Sactuel, Sinit, Max : REAL;
VAR Nom : Str20;
VAR cmap : ARRAY [0..31] OF CARDINAL;
VAR TabNombre : ARRAY [0..15] OF CARDINAL;

VAR TabNombreInit : ARRAY [0..15] OF CARDINAL;
VAR JourAct : INTEGER;
VAR TempsSommeInit, TempsSIBillets,TempsSommeRec,TempsMomentRec,TempsPosteRec
    TempsSommeDep,TempsMomentDep,TempsPosteDep,
    TempsTotal,TempsCorrection : INTEGER;
VAR TableauChaine : RECORD
                      Tab : ARRAY [0..499] OF INTEGER;
                      Indice : INTEGER;
                    END;
VAR TableauTemps : ARRAY [0..499] OF INTEGER;
VAR AncMinute,AncTick : INTEGER;
VAR Mattrans : ARRAY [0..32],[0..32] OF INTEGER;
VAR nomuser,Numseance : Str20;
END VariablesGlobales.
```

Le programme Comptes : les modules spécifiques à la version suisse

```
IMPLEMENTATION MODULE CorrPMBillets;

FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM AmigaDOS IMPORT DateStamp, DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton,AddGadgetImageButton, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech, CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
       StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100,TabNombreInit,
                              TableauTraces,TRecettes,TDepenses,
                              TableauParametres,Sinit,Max,TabNombre,Sactuel,
                              TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                              ConvStringToReal;
FROM Utilitaires IMPORT ConvPrononcable;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;
VAR
  Done,DoneOK,DoneRec,Recom : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff,QR : BOOLEAN
  Gl,GlEff,GlQui,GlFin,GlOK,GlRec : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK,WpRec : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK,ReqRec : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK,MsgRec : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  Trace : Str40;
  Der : Dernier;
  Somme, SommePrecedente, SommeMax,ValeurMax,ValeurPrec,SomAct,SA,SAP : REAL
  StrSomme,ES,fonc,SommeString : Str20;
  For : RealToStringFormat;
  Win,Won,Win2,Win3,Win4 : WindowPtr;
```

```
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  TabImages : ARRAY [0..59] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3,ImgPtr4,ImgPtr5 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,ImgCount4,ImgCount5,
                        Indice,i,I,X,Y,list6,list7 : CARDINAL;
  Res : BOOLEAN;
  SomPron,PPron,DerPron : Str100;
  Phrase,Maxaf  : Str20;
  Phrase2 : ARRAY [0..39] OF CHAR;
  In : CARDINAL;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree : REAL;
  DSR : DateStampRecord;

PROCEDURE GadgetHandlerEffBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Efface := TRUE ; DoneEff := TRUE;
           Trace := "Il confirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END ;
     1 : Efface := FALSE; DoneEff := TRUE;
           Trace := "Il infirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END ;
     2 :
  END;
END GadgetHandlerEffBS;

PROCEDURE GadgetHandlerFinBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN

  CASE Gad.GadgetID OF
     0 : Fini := TRUE ; DoneFin := TRUE;
           IF TableauChaine.Indice < 499
           THEN TableauChaine.Tab[TableauChaine.Indice] :=29;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
           TableauChaine.Indice := TableauChaine.Indice + 1;
           TableauChaine.Tab[TableauChaine.Indice] :=7;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute))
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
```

```
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il confirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |

    1 : Fini := FALSE; DoneFin := TRUE;
            IF TableauChaine.Indice < 499
            THEN TableauChaine.Tab[TableauChaine.Indice] :=29;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
                TableauChaine.Tab[TableauChaine.Indice] :=26;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END|
    2 :
  END;
END GadgetHandlerFinBS;

PROCEDURE GadgetHandlerRecBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget)
BEGIN

  CASE Gad.GadgetID OF
    0 : Recom := TRUE ; DoneRec := TRUE;
            Trace := "Il confirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END |
    1 : Recom := FALSE; DoneRec := TRUE;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END |
    2 :
```

```
    END;
END GadgetHandlerRecBS;

PROCEDURE GadgetHandlerBS (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
BEGIN
  IF OpenSpeech () THEN
  CASE Gad.GadgetID OF
     0 : ActivEff := TRUE;
         Nb100 := Nb100 + 1;
         SommePrecedente := Somme;
         Somme := Somme + 100.;
         IF Nb100 <= 5
         THEN
            Der.X := 170 + ((Nb100 - 1) * 20);
            Der.Y := 17;
            Der.Type := "BIL";
            Der.Last := 100.;
            AuMoinsUn := TRUE;
            Delay (25);
            DrawImage (Win^.RPort^,TabImages[11],Der.X,Der.Y)
         ELSE
            AuMoinsUn := FALSE;
            Der.Last := 100.
         END;
         IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("soh froh"));
           DerPron := "soh froh";
         END;
         wMove (Won^,7,1);
         wClrEndLine (Won^);
         ConvRealToString (Somme, StrSomme, 2, For);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,15,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);
         Trace := "Il appuie sur 100 Frs ";
         IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END ;
     1 : ActivEff := TRUE;
         Nb50 := Nb50 + 1;
         SommePrecedente := Somme;
         Somme := Somme + 50.;
         IF Nb50 <= 5
         THEN
            Der.X := 170 + ((Nb50 - 1) * 20);
            Der.Y := 34;
            Der.Type := "BIL";
            Der.Last := 50.;
            AuMoinsUn := TRUE;
            Delay (25);
            DrawImage (Win^.RPort^,TabImages[12],Der.X,Der.Y)
         ELSE
            AuMoinsUn := FALSE;
            Der.Last := 50.
         END;
         IF (TableauParametres[1] = 2) THEN
            Res := SayAndReturn (ADR("sekant froh"));
            DerPron := "sekant froh";
         END;
         wMove (Won^,7,1);
         wClrEndLine (Won^);
         ConvRealToString (Somme, StrSomme, 2, For);
         PutStr (Won^,ADR(StrSomme));
```

```
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 50 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
2 : ActivEff := TRUE;
    Nb20 := Nb20 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 20.;
    IF Nb20 <= 5
    THEN
       Der.X := 170 + ((Nb20 - 1) * 20);
       Der.Y := 51;
       Der.Type := "BIL";
       Der.Last := 20.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[13],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 20.
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("vant froh"));
      DerPron := "vant froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 20 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
3 : ActivEff := TRUE;
    Nb10 := Nb10 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 10.;
    IF Nb10 <= 5
    THEN
       Der.X := 170 + ((Nb10 - 1) * 20);
       Der.Y := 68;
       Der.Type := "BIL";
       Der.Last := 10.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[14],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 10.
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("dys froh"));
      DerPron := "dys froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
```

```
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 10 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  4 : ActivEff := TRUE;
      Nb5 := Nb5 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 5.;
      IF Nb5 <= 5
      THEN
         Der.X := 170 + ((Nb5 - 1) * 24);
         Der.Y := 84;
         Der.Type := "P5F";
         Der.Last := 5.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[15],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 5.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("senk froh"));
        DerPron := "senk froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 5 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  5 : ActivEff := TRUE;
      Nb2 := Nb2 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 2.;
      IF Nb2 <= 7
      THEN
         Der.X := 170 + ((Nb2 - 1) * 18);
         Der.Y := 107;
         Der.Type := "PA";
         Der.Last := 2.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[16],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 2.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("du froh"));
        DerPron := "du froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
```

```
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);

      Trace := "Il appuie sur 2 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  6 : ActivEff := TRUE;
      Nb1 := Nb1 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 1.;
      IF Nb1 <= 7
      THEN
         Der.X := 170 + ((Nb1 - 1) * 18);
         Der.Y := 125;
         Der.Type := "PA";
         Der.Last := 1.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[17],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 1.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("an froh"));
        DerPron := "an froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 1 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  7 : ActivEff := TRUE;
      Nb05 := Nb05 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.5;
      IF Nb05 <= 7
      THEN
         Der.X := 170 + ((Nb05 - 1) * 18);
         Der.Y := 143;
         Der.Type := "PA";
         Der.Last := 0.5;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[7],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 0.5
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("sekan't santeam"));
        DerPron := "sekan't santeam";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
```

```
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 0.50 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  8 : ActivEff := TRUE;
      Nb02 := Nb02 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.2;
      IF Nb02 <= 7
      THEN
         Der.X := 170 + ((Nb02 - 1) * 18);
         Der.Y := 161;
         Der.Type := "PA";
         Der.Last := 0.2;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[18],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 0.2
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("vant santeam"));
        DerPron := "vant santeam";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 0.20 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  9 : ActivEff := TRUE;
      Nb01 := Nb01 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.1;
      IF Nb01 <= 7
      THEN
         Der.X := 170 + ((Nb01 - 1) * 18);
         Der.Y := 179;
         Der.Type := "PA";
         Der.Last := 0.1;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[19],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 0.1
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("dys santeam"));
        DerPron := "dys santeam";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
```

```
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 0.10 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
10: ActivEff := TRUE;
    Nb005 := Nb005 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 0.05;
    IF Nb005 <= 7
    THEN
       Der.X := 170 + ((Nb005 - 1) * 18);
       Der.Y := 197;
       Der.Type := "PA";
       Der.Last := 0.05;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[10],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 0.05
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("senk santeam"));
      DerPron := "senk santeam";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 0.05 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
11: IF ActivEff THEN
      IF TableauChaine.Indice < 499
      THEN TableauChaine.Tab[TableauChaine.Indice] :=27;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
           TableauChaine.Indice := TableauChaine.Indice + 1;
           TableauChaine.Tab[TableauChaine.Indice] :=26;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute))
                END;
```

```
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
        TableauChaine.Indice := TableauChaine.Indice + 1;
END;
Trace := "Il demande d'effacer ";
IF TableauTraces.Indice <= 499 THEN
     CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
     TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                  RelVerify});
AddGadgetTextButton (30,34, ADR("OUI"));
AddGadgetTextButton (115,34, ADR("NON"));
AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS EFFACER ?"));
GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
           GadgetMutualExcludeSet {});
G1Eff := EndGadgetList ();
InitRequester (ReqEff);
WITH ReqEff DO
       OlderRequest := NIL;
       LeftEdge := 70;
       TopEdge := 8;
       Width := 180;
       Height := 50;
       ReqGadget := G1Eff;
       ReqBorder := NIL;
       ReqText := NIL;
       Flags := RequesterFlagsSet {};
       BackFill := BYTE(10);
       ReqLayer := NIL;
       ImageBMap := NIL;
END;
IF Request (ReqEff, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("Voolay voo effahsay ?"));
   END;
   DoneEff := FALSE;
   WHILE NOT DoneEff DO
       Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
       LOOP
          MsgEff := GetMsg (Win^.UserPort^);
          IF (MsgEff = NIL) THEN EXIT; END;
          ProcIMsg (WpEff, MsgEff);
       END;
   END;
END;
FreeGadgetList (G1Eff^);
Delay(25);
IF Efface THEN
  ActivEff := FALSE;
  IF ((CompareString (Der.Type,"BIL") = equal) AND (AuMoinsUn))
    THEN DrawImage (Win^.RPort^,TabImages[24],Der.X,Der.Y);
      IF (Der.X # 170) THEN
        IF Der.Last = 100.
          THEN DrawImage (Win^.RPort^,TabImages[11],Der.X - 20,Der
          ELSIF Der.Last = 50.
          THEN DrawImage (Win^.RPort^,TabImages[12],Der.X - 20,Der
          ELSIF Der.Last = 20.
          THEN DrawImage (Win^.RPort^,TabImages[13],Der.X - 20,Der
          ELSIF Der.Last = 10.
          THEN DrawImage (Win^.RPort^,TabImages[14],Der.X - 20,Der
```

```
            END
          END
        ELSIF ((CompareString (Der.Type,"P5F") = equal) AND (AuMoinsUn)
          THEN DrawImage (Win^.RPort^,TabImages[25],Der.X,Der.Y)
          ELSIF ((CompareString (Der.Type,"PA") = equal) AND (AuMoinsUn
            THEN DrawImage (Win^.RPort^,TabImages[26],Der.X,Der.Y)
      END;
      SommePrecedente := Somme;
      Somme := Somme - Der.Last;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      DerPron := "";
      IF Der.Last = 100.
       THEN  Nb100 := Nb100 - 1
       ELSIF  Der.Last = 50.
       THEN  Nb50 := Nb50 - 1
       ELSIF  Der.Last = 20.
       THEN  Nb20 := Nb20 - 1
       ELSIF  Der.Last = 10.
       THEN  Nb10 := Nb10 - 1
       ELSIF  Der.Last = 5.
       THEN  Nb5 := Nb5 - 1
       ELSIF  Der.Last = 2.
       THEN  Nb2 := Nb2 - 1
       ELSIF  Der.Last = 1.
       THEN  Nb1 := Nb1 - 1
       ELSIF  Der.Last = 0.5
       THEN  Nb05 := Nb05 - 1
       ELSIF  Der.Last = 0.2
       THEN  Nb02 := Nb02 - 1
       ELSIF  Der.Last = 0.1
       THEN  Nb01 := Nb01 - 1
       ELSIF Der.Last = 0.05
       THEN  Nb005 := Nb005 - 1
       ELSE
      END;
    END
  END ;

12 :Trace := "Il appuie sur OK ";
  IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
  END;
  BeginGadgetList ();
  GadgetTypeReq := TRUE;
  GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                   RelVerify});
  AddGadgetTextButton (30,34, ADR("OUI"));
  AddGadgetTextButton (115,34, ADR("NON"));
  AddGadgetTextButton (15,9, ADR("EST-CE BIEN JUSTE ?"));
  GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
             GadgetMutualExcludeSet {});
  GlFin := EndGadgetList ();
  InitRequester (ReqFin);
  WITH ReqFin DO
        OlderRequest := NIL;
        LeftEdge := 70;
        TopEdge := 8;
        Width := 180;
        Height := 50;
```

```
            ReqGadget := GlFin;
            ReqBorder := NIL;
            ReqText := NIL;
            Flags := RequesterFlagsSet {};
            BackFill := BYTE(10);
            ReqLayer := NIL;
            ImageBMap := NIL;
      END;
      IF Request (ReqFin, Win^) THEN
         IF (TableauParametres[1] = 2) THEN
            Res := SayAndReturn (ADR("ahvay voo feeny ?"));
         END;
         DoneFin := FALSE;
         WHILE NOT DoneFin DO
            Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
            LOOP
               MsgFin := GetMsg (Win^.UserPort^);
               IF (MsgFin = NIL) THEN EXIT; END;
               ProcIMsg (WpFin, MsgFin);
            END;
         END;
      END;
      FreeGadgetList (GlFin^);
      Delay(25);
      IF Fini THEN Done := TRUE END;

13:   IF TableauChaine.Indice < 499
      THEN TableauChaine.Tab[TableauChaine.Indice] :=28;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
            TableauChaine.Tab[TableauChaine.Indice] :=26;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
               ActTick2 := ActTick;
               IF ActMinute > AncMinute
               THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
               END;
               ent := ActTick - AncTick;
               ree := real(ent);
               TableauTemps [TableauChaine.Indice] := entier(ree/50.);
               AncMinute := ActMinute;
               AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
      END;
      Trace := "Il appuie sur RECOMMENCER ";
      IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      BeginGadgetList ();
      GadgetTypeReq := TRUE;
      GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                       RelVerify});
      AddGadgetTextButton (30,34, ADR("OUI"));
```

```
AddGadgetTextButton (115,34, ADR("NON "));
AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS RECOMMENCER ?"));
GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
        GadgetMutualExcludeSet {});
GlRec := EndGadgetList ();
InitRequester (ReqRec);
WITH ReqRec DO
     OlderRequest := NIL;
     LeftEdge := 50;
     TopEdge := 8;
     Width := 220;
     Height := 50;
     ReqGadget := GlRec;
     ReqBorder := NIL;
     ReqText := NIL;
     Flags := RequesterFlagsSet {};
     BackFill := BYTE(10);
     ReqLayer := NIL;
     ImageBMap := NIL;
END;
IF Request (ReqRec, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("Vooley voo ra commohsay ?"));
   END;
   DoneRec := FALSE;
   WHILE NOT DoneRec DO
     Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
     LOOP
        MsgRec := GetMsg (Win^.UserPort^);
        IF (MsgRec = NIL) THEN EXIT; END;
        ProcIMsg (WpRec, MsgRec);
     END;
 END;
END;
FreeGadgetList (GlRec^);
Delay(25);
IF Recom THEN
     DrawImage (Win^.RPort^,TabImages[24],170,17);
     DrawImage (Win^.RPort^,TabImages[24],190,17);
     DrawImage (Win^.RPort^,TabImages[24],210,17);
     DrawImage (Win^.RPort^,TabImages[24],230,17);
     DrawImage (Win^.RPort^,TabImages[24],250,17);
     DrawImage (Win^.RPort^,TabImages[24],170,34);
     DrawImage (Win^.RPort^,TabImages[24],190,34);
     DrawImage (Win^.RPort^,TabImages[24],210,34);
     DrawImage (Win^.RPort^,TabImages[24],230,34);
     DrawImage (Win^.RPort^,TabImages[24],250,34);
     DrawImage (Win^.RPort^,TabImages[24],170,51);
     DrawImage (Win^.RPort^,TabImages[24],190,51);
     DrawImage (Win^.RPort^,TabImages[24],210,51);
     DrawImage (Win^.RPort^,TabImages[24],230,51);
     DrawImage (Win^.RPort^,TabImages[24],250,51);
     DrawImage (Win^.RPort^,TabImages[24],170,68);
     DrawImage (Win^.RPort^,TabImages[24],190,68);
     DrawImage (Win^.RPort^,TabImages[24],210,68);
     DrawImage (Win^.RPort^,TabImages[24],230,68);
     DrawImage (Win^.RPort^,TabImages[24],250,68);
     DrawImage (Win^.RPort^,TabImages[25],170,84);
     DrawImage (Win^.RPort^,TabImages[25],194,84);
     DrawImage (Win^.RPort^,TabImages[25],218,84);
     DrawImage (Win^.RPort^,TabImages[25],242,84);
     DrawImage (Win^.RPort^,TabImages[25],266,84);
     DrawImage (Win^.RPort^,TabImages[26],170,107);
     DrawImage (Win^.RPort^,TabImages[26],188,107);
     DrawImage (Win^.RPort^,TabImages[26],206,107);
     DrawImage (Win^.RPort^,TabImages[26],224,107);
```

```
DrawImage (Win^.RPort^,TabImages[26],242,107);
DrawImage (Win^.RPort^,TabImages[26],260,107);
DrawImage (Win^.RPort^,TabImages[26],278,107);
DrawImage (Win^.RPort^,TabImages[26],170,125);
DrawImage (Win^.RPort^,TabImages[26],188,125);
DrawImage (Win^.RPort^,TabImages[26],206,125);
DrawImage (Win^.RPort^,TabImages[26],224,125);
DrawImage (Win^.RPort^,TabImages[26],242,125);
DrawImage (Win^.RPort^,TabImages[26],260,125);
DrawImage (Win^.RPort^,TabImages[26],278,125);
DrawImage (Win^.RPort^,TabImages[26],170,143);
DrawImage (Win^.RPort^,TabImages[26],188,143);
DrawImage (Win^.RPort^,TabImages[26],206,143);
DrawImage (Win^.RPort^,TabImages[26],224,143);
DrawImage (Win^.RPort^,TabImages[26],242,143);
DrawImage (Win^.RPort^,TabImages[26],260,143);
DrawImage (Win^.RPort^,TabImages[26],278,143);
DrawImage (Win^.RPort^,TabImages[26],170,161);
DrawImage (Win^.RPort^,TabImages[26],188,161);
DrawImage (Win^.RPort^,TabImages[26],206,161);
DrawImage (Win^.RPort^,TabImages[26],224,161);
DrawImage (Win^.RPort^,TabImages[26],242,161);
DrawImage (Win^.RPort^,TabImages[26],260,161);
DrawImage (Win^.RPort^,TabImages[26],278,161);
DrawImage (Win^.RPort^,TabImages[26],170,179);
DrawImage (Win^.RPort^,TabImages[26],188,179);
DrawImage (Win^.RPort^,TabImages[26],206,179);
DrawImage (Win^.RPort^,TabImages[26],224,179);
DrawImage (Win^.RPort^,TabImages[26],242,179);
DrawImage (Win^.RPort^,TabImages[26],260,179);
DrawImage (Win^.RPort^,TabImages[26],278,179);
DrawImage (Win^.RPort^,TabImages[26],170,197);
DrawImage (Win^.RPort^,TabImages[26],188,197);
DrawImage (Win^.RPort^,TabImages[26],206,197);
DrawImage (Win^.RPort^,TabImages[26],224,197);
DrawImage (Win^.RPort^,TabImages[26],242,197);
DrawImage (Win^.RPort^,TabImages[26],260,197);
DrawImage (Win^.RPort^,TabImages[26],278,197);
Somme := 0.;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);
ES := "";
SommeMax := Max;
Nb100 := 0;
Nb50 := 0;
Nb20 := 0;
Nb10 := 0;
Nb5 := 0;
Nb2 := 0;
Nb1 := 0;
Nb05 := 0;
Nb02 := 0;
Nb01 := 0;
Nb005 := 0;
Der.Last := 0.;
Der.X := 150;
AuMoinsUn := FALSE;
ActivEff := FALSE;
For := Decimal;
CopyString (DerPron,PPron);
```

```
            END;

      14:   IF (TableauParametres[1] = 2) THEN
              Res := SayAndReturn (ADR(DerPron));
            END;
  END;
CloseSpeech ();
END;
END GadgetHandlerBS;


PROCEDURE CorrPMBI (VAR Phrase : ARRAY OF CHAR;
                    VAR PhrasePron : ARRAY OF CHAR );


BEGIN
    IF TableauChaine.Indice <= 499
    THEN TableauChaine.Tab[TableauChaine.Indice] :=26;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
        TableauChaine.Indice := TableauChaine.Indice + 1;
    END;

    CopyString (PPron,PhrasePron);
    QR := FALSE;
    Somme := 0.;
    SommeMax := Max;
    Nb100 := 0;
    Nb50 := 0;
    Nb20 := 0;
    Nb10 := 0;
    Nb5 := 0;
    Nb2 := 0;
    Nb1 := 0;
    Nb05 := 0;
    Nb02 := 0;
    Nb01 := 0;
    Nb005 := 0;
    Der.Last := 0.;
    Der.X := 150;
    AuMoinsUn := FALSE;
    ActivEff := FALSE;
    For := Decimal;

(*    IF OpenSpeech () THEN *)
    WITH Wp DO
      procGadgetUp := GadgetHandlerBS;
    END;
    WITH WpEff DO
      procGadgetUp := GadgetHandlerEffBS;
    END;
    WITH WpFin DO
      procGadgetUp := GadgetHandlerFinBS;
    END;
    WITH WpRec DO
      procGadgetUp := GadgetHandlerRecBS;
```

```
END;
Scr := CreateScreen (320,230,5,NIL);
IF (Scr # NIL) THEN
  cmap[00] := 0000H;
  cmap[01] := 0FFFH;
  cmap[02] := 0E00H;
  cmap[03] := 0A00H;
  cmap[04] := 0D80H;
  cmap[05] := 0FE0H;
  cmap[06] := 0FCAH;
  cmap[07] := 0080H;
  cmap[08] := 00B6H;
  cmap[09] := 00DDH;
  cmap[10] := 00AFH;
  cmap[11] := 007CH;
  cmap[12] := 000FH;
  cmap[13] := 070FH;
  cmap[14] := 0C0EH;
  cmap[15] := 0C08H;
  cmap[16] := 0620H;
  cmap[17] := 0E52H;
  cmap[18] := 0A52H;
  cmap[19] := 0FCAH;
  cmap[20] := 0333H;
  cmap[21] := 0444H;
  cmap[22] := 0555H;
  cmap[23] := 0666H;
  cmap[24] := 0777H;
  cmap[25] := 0888H;
  cmap[26] := 0999H;
  cmap[27] := 0AAAH;
  cmap[28] := 0CCCH;
  cmap[29] := 0DDDH;
  cmap[30] := 0EEEH;
  cmap[31] := 0FFFH;
  LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

  ImgPtr := FindImageTable(33333333H, ImgCount);
  IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
    Indice := 0;
    WHILE (Indice <= ImgCount-1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr^[Indice].Width;
        Height      := ImgPtr^[Indice].Height;
        Depth       := ImgPtr^[Indice].Depth;
        ImageData   := ImgPtr^[Indice].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice := Indice + 1;
    END;

 ImgPtr2 := FindImageTable(44444444H, ImgCount2);
  IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
   WHILE (Indice-ImgCount <= ImgCount2-1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr2^[Indice-ImgCount].Width;
        Height      := ImgPtr2^[Indice-ImgCount].Height;
        Depth       := ImgPtr2^[Indice-ImgCount].Depth;
        ImageData   := ImgPtr2^[Indice-ImgCount].Data;
        PlanePick   := BYTE(31);
```

```
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      Indice := Indice + 1;
    END;
    ImgPtr3 := FindImageTable(77777777H, ImgCount3);
    IF (ImgPtr3 # NIL) AND (ImgCount3 # 0) THEN
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr3^[9].Width;
        Height      := ImgPtr3^[9].Height;
        Depth       := ImgPtr3^[9].Depth;
        ImageData   := ImgPtr3^[9].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
     END;
     Indice := Indice + 1;

 ImgPtr4 := FindImageTable(66666666H, ImgCount4);
 IF (ImgPtr4 # NIL) AND (ImgCount4 # 0) THEN
   list6 := 3;
   WHILE (list6 <= ImgCount4-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr4^[list6].Width;
       Height      := ImgPtr4^[list6].Height;
       Depth       := ImgPtr4^[list6].Depth;
       ImageData   := ImgPtr4^[list6].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list6 := list6 + 1;
     Indice := Indice + 1;
   END;
ImgPtr5 := FindImageTable(11111114H, ImgCount5);
 IF (ImgPtr5 # NIL) AND (ImgCount5 # 0) THEN
  list7 := 3;
  WHILE (list7 <= ImgCount5-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr5^[list7].Width;
       Height      := ImgPtr5^[list7].Height;
       Depth       := ImgPtr5^[list7].Depth;
       ImageData   := ImgPtr5^[list7].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list7 := list7 + 1;
     Indice := Indice + 1;
   END;


    BeginGadgetList();
    AddGadgetImageButton (10,10,TabImages[0]);
    AddGadgetImageButton (10,47,TabImages[1]);
    AddGadgetImageButton (10,84,TabImages[2]);
    AddGadgetImageButton (10,121,TabImages[3]);
    AddGadgetImageButton (80,5,TabImages[4]);
    AddGadgetImageButton (85,45,TabImages[5]);
```

```
AddGadgetImageButton (87,78,TabImages[5]);
AddGadgetImageButton (90,109,TabImages[7]);
AddGadgetImageButton (90,132,TabImages[8]);
AddGadgetImageButton (91,156,TabImages[9]);
AddGadgetImageButton (92,178,TabImages[10]);
AddGadgetImageButton (4,200,TabImages[20]);
AddGadgetImageButton (121,200,TabImages[21]);
AddGadgetImageButton (63,200,TabImages[36]);
IF TableauParametres[1] = 2 THEN
   AddGadgetImageButton (6,160,TabImages[35]);
END;
Gl := EndGadgetList();
IF (Gl # NIL) THEN
   Win := CreateWindow (0,0,320,230,WIDCMP,WFlags,Gl,Scr,NIL);
   Win2 := CreateWindow (134,72,23,102,WIDCMP,WFlags2,NIL,Scr,NIL);

   Won := CreateWindow (168,212,152,15,WIDCMP,WFlags,NIL,Scr,NIL);
   IF (Win # NIL) AND (Won # NIL)  THEN

   IF CreateConsole (Won^) THEN
      wClrScr (Won^);
      PutStr (Won^,ADR("Total"));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      DrawImage (Win^.RPort^,TabImages[28],165,0);
      DrawImage (Win^.RPort^,TabImages[28],165,100);
      IF NOT QR THEN
        DrawImage (Win^.RPort^,TabImages[43],0,0);
        DrawImage (Win^.RPort^,TabImages[43],130,0);
        DrawImage (Win^.RPort^,TabImages[43],190,0);
        DrawImage (Win^.RPort^,TabImages[43],0,227);
        DrawImage (Win^.RPort^,TabImages[43],130,227);
        DrawImage (Win^.RPort^,TabImages[43],190,227);
        DrawImage (Win^.RPort^,TabImages[45],0,0);
        DrawImage (Win^.RPort^,TabImages[45],0,100);
        DrawImage (Win^.RPort^,TabImages[45],317,0);
        DrawImage (Win^.RPort^,TabImages[45],317,100);
        DrawImage (Win^.RPort^,TabImages[42],123,3);
        DrawImage (Win^.RPort^,TabImages[45],165,0);
        DrawImage (Win^.RPort^,TabImages[45],165,100);
        DrawImage (Win^.RPort^,TabImages[39],175,5);

      END;
      IF (CompareString (PhrasePron,"Cobya ah vay voo ra su darjo ?"
              = equal)
      THEN
        DrawImage (Win^.RPort^,TabImages[31],0,0);
        DrawImage (Win^.RPort^,TabImages[31],130,0);
        DrawImage (Win^.RPort^,TabImages[31],190,0);
        DrawImage (Win^.RPort^,TabImages[31],0,227);
        DrawImage (Win^.RPort^,TabImages[31],130,227);
        DrawImage (Win^.RPort^,TabImages[31],190,227);
        DrawImage (Win^.RPort^,TabImages[29],0,0);
        DrawImage (Win^.RPort^,TabImages[29],0,100);
        DrawImage (Win^.RPort^,TabImages[29],317,0);
        DrawImage (Win^.RPort^,TabImages[29],317,100);
        DrawImage (Win^.RPort^,TabImages[33],123,3);
        DrawImage (Win^.RPort^,TabImages[29],165,0);
        DrawImage (Win^.RPort^,TabImages[29],165,100);
        DrawImage (Win^.RPort^,TabImages[41],175,5);

      ELSIF (CompareString (PhrasePron,"Cobya ah vay voo daypensay ?
              = equal)
      THEN
        DrawImage (Win^.RPort^,TabImages[32],0,0);
```

```
          DrawImage (Win^.RPort^,TabImages[32],130,0);
          DrawImage (Win^.RPort^,TabImages[32],190,0);
          DrawImage (Win^.RPort^,TabImages[32],0,227);
          DrawImage (Win^.RPort^,TabImages[32],130,227);
          DrawImage (Win^.RPort^,TabImages[32],190,227);
          DrawImage (Win^.RPort^,TabImages[30],0,0);
          DrawImage (Win^.RPort^,TabImages[30],0,100);
          DrawImage (Win^.RPort^,TabImages[30],317,0);
          DrawImage (Win^.RPort^,TabImages[30],317,100);
          DrawImage (Win^.RPort^,TabImages[34],123,3);
          DrawImage (Win^.RPort^,TabImages[30],165,0);
          DrawImage (Win^.RPort^,TabImages[30],165,100);
          DrawImage (Win^.RPort^,TabImages[40],175,5);

        END;
        IF TableauParametres[1] = 2 THEN
          IF OpenSpeech () THEN
                Res := SayAndReturn (ADR(PhrasePron));
                CopyString (DerPron,PhrasePron);
                CloseSpeech ()
          END;
        END;

        SA := Sactuel;

      In :=1;

      WHILE (In <= TabNombreInit [0]) DO

Nb100 := Nb100 + 1;
SommePrecedente := Somme;
Somme := Somme + 100.;
IF Nb100 <= 5
THEN
   Der.X := 170 + ((Nb100 - 1) * 20);
   Der.Y := 17;
   Der.Type := "BIL";
   Der.Last := 100.;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[11],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 100.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
END;

      In :=1;
      WHILE (In <= TabNombreInit [1]) DO

Nb50 := Nb50 + 1;
SommePrecedente := Somme;
Somme := Somme + 50.;
IF Nb50 <= 5
THEN
   Der.X := 170 + ((Nb50 - 1) * 20);
   Der.Y := 34;
```

```
   Der.Type := "BIL";
   Der.Last := 50.;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[12],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 50.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [2]) DO

Nb20 := Nb20 + 1;
SommePrecedente := Somme;
Somme := Somme + 20.;
IF Nb20 <= 5
THEN
   Der.X := 170 + ((Nb20 - 1) * 20);
   Der.Y := 51;
   Der.Type := "BIL";
   Der.Last := 20.;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[13],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 20.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);


   In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [3]) DO


Nb10 := Nb10 + 1;
SommePrecedente := Somme;
Somme := Somme + 10.;
IF Nb10 <= 5
THEN
   Der.X := 170 + ((Nb10 - 1) * 20);
   Der.Y := 68;
   Der.Type := "BIL";
   Der.Last := 10.;
   AuMoinsUn := TRUE;
```

```
   DrawImage (Win^.RPort^,TabImages[14],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 10.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);


   In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [4]) DO

Nb5 := Nb5 + 1;
SommePrecedente := Somme;
Somme := Somme + 5.;
IF Nb5 <= 5
THEN
   Der.X := 170 + ((Nb5 - 1) * 24);
   Der.Y := 84;
   Der.Type := "P5F";
   Der.Last := 5.;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[15],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 5.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [5]) DO

Nb2 := Nb2 + 1;
SommePrecedente := Somme;
Somme := Somme + 2.;
IF Nb2 <= 7
THEN
   Der.X := 170 + ((Nb2 - 1) * 18);
   Der.Y := 107;
   Der.Type := "PA";
   Der.Last := 2.;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[16],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 2.
END;
```

```
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);



   In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [6]) DO

Nb1 := Nb1 + 1;
SommePrecedente := Somme;
Somme := Somme + 1.;
IF Nb1 <= 7
THEN
   Der.X := 170 + ((Nb1 - 1) * 18);
   Der.Y := 125;
   Der.Type := "PA";
   Der.Last := 1.;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[17],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 1.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [7]) DO

Nb05 := Nb05 + 1;
SommePrecedente := Somme;
Somme := Somme + 0.5;
IF Nb05 <= 7
THEN
   Der.X := 170 + ((Nb05 - 1) * 18);
   Der.Y := 143;
   Der.Type := "PA";
   Der.Last := 0.5;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[7],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 0.5
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
```

```
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [8]) DO

NbO2 := NbO2 + 1;
SommePrecedente := Somme;
Somme := Somme + 0.2;
IF NbO2 <= 7
THEN
   Der.X := 170 + ((NbO2 - 1) * 18);
   Der.Y := 161;
   Der.Type := "PA";
   Der.Last := 0.2;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[18],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 0.2
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [9]) DO

NbO1 := NbO1 + 1;
SommePrecedente := Somme;
Somme := Somme + 0.1;
IF NbO1 <= 7
THEN
   Der.X := 170 + ((NbO1 - 1) * 18);
   Der.Y := 179;
   Der.Type := "PA";
   Der.Last := 0.1;
   AuMoinsUn := TRUE;

   DrawImage (Win^.RPort^,TabImages[19],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 0.1
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
END;
```

```
            In :=1;
            WHILE (In <= TabNombreInit [10]) DO

        Nb005 := Nb005 + 1;
        SommePrecedente := Somme;
        Somme := Somme + 0.05;
        IF Nb005 <= 7
        THEN
           Der.X := 170 + ((Nb005 - 1) * 18);
           Der.Y := 197;
           Der.Type := "PA";
           Der.Last := 0.05;
           AuMoinsUn := TRUE;

           DrawImage (Win^.RPort^,TabImages[10],Der.X,Der.Y)
        ELSE
           AuMoinsUn := FALSE;
           Der.Last := 0.05
        END;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 2, For);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);

           In := In+1;
        END;

            Done := FALSE;
            WHILE (NOT Done) DO
              Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
              LOOP
                Msg := GetMsg(Win^.UserPort^);
                IF (Msg = NIL) THEN EXIT; END;
                ProcIMsg (Wp, Msg);
              END;
            END;
            DeleteConsole (Won^);
            ELSE WriteString ("Console non créée")
          END;
        Sinit := Somme;

        CloseWindow(Won^);
        CloseWindow(Win^);
        END;
        FreeGadgetList (Gl^);
      END;
    CloseScreen(Scr^);
  END;
 END;
END;
END;

  TabNombreInit[0] := Nb100;
  TabNombreInit[1] := Nb50;
  TabNombreInit[2] := Nb20;
  TabNombreInit[3] := Nb10;
  TabNombreInit[4] := Nb5;
  TabNombreInit[5] := Nb2;
  TabNombreInit[6] := Nb1;
  TabNombreInit[7] := Nb05;
  TabNombreInit[8] := Nb02;
```

```
  TabNombreInit[9] := Nb01;
  TabNombreInit[10] := Nb005;

END CorrPMBI;



(*  VAR p,pp : ARRAY [0..40] OF CHAR;

BEGIN
    p := "Combien avez-vous d'argent ?";
    pp := "Cobya ah vay voo darjo ?";
    CorrPMBI (p,pp); *)


END CorrPMBillets.
```

```
IMPLEMENTATION MODULE SSBillets;

FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request,RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton,AddGadgetImageButton, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
       StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100,TabNombreInit,
                TempsSIBillets,TableauTraces,TRecettes,TDepenses,
                               TableauParametres,Sinit,Max,TabNombre,Sactuel,
                               TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                               ConvStringToReal;
FROM Utilitaires IMPORT ConvPrononcable;

CONST
  WIDCMP = IDCMPFlagsSet éGadgetUpè;
  WFlags = WindowFlagsSet éActivateè;
  WFlags2 = WindowFlagsSet éActivate,Borderlessè;
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;
VAR
  Done,DoneOK,DoneRec,Recom : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff,QR : BOOLEAN
  Gl,GlEff,GlQui,GlFin,GlOK,GlRec : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK,WpRec : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK,ReqRec : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK,MsgRec : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  Trace : Str40;
  Der : Dernier;
  Somme, SommePrecedente, SommeMax,ValeurMax,ValeurPrec,SomAct,SA,SAP : REAL;
  StrSomme,ES,fonc,SommeString : Str20;
  For : RealToStringFormat;
  Win,Won,Win2,Win3,Win4 : WindowPtr;
```

```
  Scr : ScreenPtr;
  cmap : ARRAY °0..31§ OF CARDINAL;
  TabImages : ARRAY °0..59§ OF Image;
  ImgPtr,ImgPtr2,ImgPtr3,ImgPtr4,ImgPtr5 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,ImgCount4,ImgCount5,
                        Indice,i,I,X,Y,list6,list7 : CARDINAL;
  Res : BOOLEAN;
  SomPron,PPron,DerPron : Str100;
  Phrase,Maxaf  : Str20;
  Phrase2 : ARRAY °0..39§ OF CHAR;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree,ree2 : REAL;
  DSR : DateStampRecord;
  IndTemps,II : CARDINAL;
  NminBEG, NticksBEG,NticksEND,NminEND,TempsInterm : ARRAY °1..20§ OF INTEGER;
  Nbillets : INTEGER;

PROCEDURE GadgetHandlerEffBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Efface := TRUE ; DoneEff := TRUE;
           IF TableauChaine.Indice <= 499
           THEN IF NOT QR
                THEN TableauChaine.Tab°TableauChaine.Indice§ := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                           = equal)
                     THEN TableauChaine.Tab °TableauChaine.Indice§ := 20;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                          TableauChaine.Indice := TableauChaine.Indice + 1;
                     ELSE TableauChaine.Tab °TableauChaine.Indice§ := 12;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
```

```
                    TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
            END;
        END;
        Trace := "Il confirme la demande ";
        IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ù
  1 : Efface := FALSE; DoneEff := TRUE;
        IF TableauChaine.Indice <= 499
        THEN IF NOT QR
            THEN TableauChaine.Tab°TableauChaine.Indice§ := 3;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
            ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                        = equal)
                 THEN TableauChaine.Tab °TableauChaine.Indice§ := 20;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                 ELSE TableauChaine.Tab °TableauChaine.Indice§ := 12;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                 END;
            END;
        END;
        Trace := "Il infirme la demande ";
        IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ù
  2 :
END;
```

```
END GadgetHandlerEffBS;

PROCEDURE GadgetHandlerQuiBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Quitter := TRUE ; DoneQui := TRUE;
         IF TableauChaine.Indice <= 499
         THEN TableauChaine.Tab°TableauChaine.Indice§ := 7;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
              TableauChaine.Indice := TableauChaine.Indice + 1;
         END;
         Trace := "Il confirme la demande ";
         IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END ù
    1 : Quitter := FALSE; DoneQui := TRUE;
         IF TableauChaine.Indice <= 499
         THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                        = equal)
              THEN TableauChaine.Tab °TableauChaine.Indice§ := 20;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
              ELSE TableauChaine.Tab °TableauChaine.Indice§ := 12;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
              END;
          END;
         Trace := "Il infirme la demande ";
         IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END ù
```

```
    2 :
  END;
END GadgetHandlerQuiBS;

PROCEDURE GadgetHandlerFinBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN

  CASE Gad.GadgetID OF
     0 : Fini := TRUE ; DoneFin := TRUE;
         IF TableauChaine.Indice <= 499
         THEN TableauChaine.Tab°TableauChaine.Indice§ := 7;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
              TableauChaine.Indice := TableauChaine.Indice + 1;
         END;
         Trace := "Il confirme la demande ";
         IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END; ù

     1 : Fini := FALSE; DoneFin := TRUE;
         IF TableauChaine.Indice <= 499
         THEN IF NOT QR
              THEN TableauChaine.Tab°TableauChaine.Indice§ := 3;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
              ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                           = equal)
                   THEN TableauChaine.Tab °TableauChaine.Indice§ := 20;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute))
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                        TableauChaine.Indice := TableauChaine.Indice + 1;
                   ELSE TableauChaine.Tab °TableauChaine.Indice§ := 12;
                  DateStamp(DSR);
```

```
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                          TableauChaine.Indice := TableauChaine.Indice + 1;
                     END;
               END;
          END;
          Trace := "Il infirme la demande ";
          IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
          ENDù
     2 :
  END;
END GadgetHandlerFinBS;
PROCEDURE GadgetHandlerRecBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN

  CASE Gad.GadgetID OF
     0 : Recom := TRUE ; DoneRec := TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab°TableauChaine.Indice§ := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
          Trace := "Il confirme la demande ";
          IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace)
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
          END ù
     1 : Recom := FALSE; DoneRec := TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab°TableauChaine.Indice§ := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
```

```
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END ù
      2 :
    END;
END GadgetHandlerRecBS;

PROCEDURE GadgetHandlerBS (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR PS: POINTER TO ARRAY °0..19§ OF CHAR;
BEGIN
  IF OpenSpeech () THEN
  CASE Gad.GadgetID OF
     0 : DateStamp(DSR);
         NminEND°IndTemps§ := SHORT(DSR.dsMinute);
         NticksEND°IndTemps§ := SHORT(DSR.dsTick);
         Nbillets := Nbillets + 1;
         ActivEff := TRUE;
         Nb100 := Nb100 + 1;
         SommePrecedente := Somme;
         Somme := Somme + 100.;
         IF Nb100 <= 5
         THEN
            Der.X := 170 + ((Nb100 - 1) * 20);
            Der.Y := 17;
            Der.Type := "BIL";
            Der.Last := 100.;
            AuMoinsUn := TRUE;
            Delay (25);
            DrawImage (Win^.RPort^,TabImages°11§,Der.X,Der.Y)
         ELSE
            AuMoinsUn := FALSE;
            Der.Last := 100.
         END;
         IF (TableauParametres°1§ = 2) THEN
           Res := SayAndReturn (ADR("soh froh"));
           DerPron := "soh froh";
         END;
         wMove (Won^,7,1);
         wClrEndLine (Won^);
         ConvRealToString (Somme, StrSomme, 2, For);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,15,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);
         Trace := "Il appuie sur 100 Frs ";
         IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END ù
     1 : DateStamp(DSR);
         NminEND°IndTemps§ := SHORT(DSR.dsMinute);
         NticksEND°IndTemps§ := SHORT(DSR.dsTick);
         Nbillets := Nbillets + 1;
         ActivEff := TRUE;
         Nb50 := Nb50 + 1;
         SommePrecedente := Somme;
         Somme := Somme + 50.;
         IF Nb50 <= 5
         THEN
            Der.X := 170 + ((Nb50 - 1) * 20);
            Der.Y := 34;
            Der.Type := "BIL";
            Der.Last := 50.;
            AuMoinsUn := TRUE;
```

```
      Delay (25);
      DrawImage (Win^.RPort^,TabImages°12§,Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 50.
   END;
   IF (TableauParametres°1§ = 2) THEN
      Res := SayAndReturn (ADR("sekant froh"));
      DerPron := "sekant froh";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 2, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);
   Trace := "Il appuie sur 50 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END ù
2 : DateStamp(DSR);
   NminEND°IndTemps§ := SHORT(DSR.dsMinute);
   NticksEND°IndTemps§ := SHORT(DSR.dsTick);
   Nbillets := Nbillets + 1;
   ActivEff := TRUE;
   Nb20 := Nb20 + 1;
   SommePrecedente := Somme;
   Somme := Somme + 20.;
   IF Nb20 <= 5
   THEN
      Der.X := 170 + ((Nb20 - 1) * 20);
      Der.Y := 51;
      Der.Type := "BIL";
      Der.Last := 20.;
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,TabImages°13§,Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 20.
   END;
   IF (TableauParametres°1§ = 2) THEN
     Res := SayAndReturn (ADR("vant froh"));
     DerPron := "vant froh";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 2, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);
   Trace := "Il appuie sur 20 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace)
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END ù
3 : DateStamp(DSR);
   NminEND°IndTemps§ := SHORT(DSR.dsMinute);
   NticksEND°IndTemps§ := SHORT(DSR.dsTick);
   Nbillets := Nbillets + 1;
   ActivEff := TRUE;
   Nb10 := Nb10 + 1;
   SommePrecedente := Somme;
```

```
      Somme := Somme + 10.;
      IF Nb10 <= 5
      THEN
         Der.X := 170 + ((Nb10 - 1) * 20);
         Der.Y := 68;
         Der.Type := "BIL";
         Der.Last := 10.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages°14§,Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 10.
      END;
      IF (TableauParametres°1§ = 2) THEN
        Res := SayAndReturn (ADR("dys froh"));
        DerPron := "dys froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 10 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ù
4 :   DateStamp(DSR);
      NminEND°IndTemps§ := SHORT(DSR.dsMinute);
      NticksEND°IndTemps§ := SHORT(DSR.dsTick);
      Nbillets := Nbillets + 1;
      ActivEff := TRUE;
      Nb5 := Nb5 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 5.;
      IF Nb5 <= 5
      THEN
         Der.X := 170 + ((Nb5 - 1) * 24);
         Der.Y := 84;
         Der.Type := "P5F";
         Der.Last := 5.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages°15§,Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 5.
      END;
      IF (TableauParametres°1§ = 2) THEN
        Res := SayAndReturn (ADR("senk froh"));
        DerPron := "senk froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 5 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
```

```
    END ù
5 : DateStamp(DSR);
    NminEND°IndTemps§ := SHORT(DSR.dsMinute);
    NticksEND°IndTemps§ := SHORT(DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb2 := Nb2 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 2.;
    IF Nb2 <= 7
    THEN
       Der.X := 170 + ((Nb2 - 1) * 18);
       Der.Y := 107;
       Der.Type := "PA";
       Der.Last := 2.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages°16§,Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 2.
    END;
    IF (TableauParametres°1§ = 2) THEN
      Res := SayAndReturn (ADR("du froh"));
      DerPron := "du froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);

    Trace := "Il appuie sur 2 Frs ";
    IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ù
6 : DateStamp(DSR);
    NminEND°IndTemps§ := SHORT(DSR.dsMinute);
    NticksEND°IndTemps§ := SHORT(DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb1 := Nb1 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 1.;
    IF Nb1 <= 7
    THEN
       Der.X := 170 + ((Nb1 - 1) * 18);
       Der.Y := 125;
       Der.Type := "PA";
       Der.Last := 1.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages°17§,Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 1.
    END;
    IF (TableauParametres°1§ = 2) THEN
      Res := SayAndReturn (ADR("an froh"));
      DerPron := "an froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
```

```
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);

      Trace := "Il appuie sur 1 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ù
  7 : DateStamp(DSR);
      NminEND°IndTemps§ := SHORT(DSR.dsMinute);
      NticksEND°IndTemps§ := SHORT(DSR.dsTick);
      Nbillets := Nbillets + 1;
      ActivEff := TRUE;
      Nb05 := Nb05 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.5;
      IF Nb05 <= 7
      THEN
         Der.X := 170 + ((Nb05 - 1) * 18);
         Der.Y := 143;
         Der.Type := "PA";
         Der.Last := 0.5;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages°7§,Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 0.5
      END;
      IF (TableauParametres°1§ = 2) THEN
        Res := SayAndReturn (ADR("sekan't santeam"));
        DerPron := "sekan't santeam";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 0.50 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ù
  8 : DateStamp(DSR);
      NminEND°IndTemps§ := SHORT(DSR.dsMinute);
      NticksEND°IndTemps§ := SHORT(DSR.dsTick);
      Nbillets := Nbillets + 1;
      ActivEff := TRUE;
      Nb02 := Nb02 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.2;
      IF Nb02 <= 7
      THEN
         Der.X := 170 + ((Nb02 - 1) * 18);
         Der.Y := 161;
         Der.Type := "PA";
         Der.Last := 0.2;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages°18§,Der.X,Der.Y)
      ELSE
```

```
        AuMoinsUn := FALSE;
        Der.Last := 0.2
    END;
    IF (TableauParametres°1§ = 2) THEN
      Res := SayAndReturn (ADR("vant santeam"));
      DerPron := "vant santeam";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 0.20 Frs ";
    IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ù
9 : DateStamp(DSR);
    NminEND°IndTemps§ := SHORT(DSR.dsMinute);
    NticksEND°IndTemps§ := SHORT(DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb01 := Nb01 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 0.1;
    IF Nb01 <= 7
    THEN
       Der.X := 170 + ((Nb01 - 1) * 18);
       Der.Y := 179;
       Der.Type := "PA";
       Der.Last := 0.1;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages°19§,Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 0.1
    END;
    IF (TableauParametres°1§ = 2) THEN
      Res := SayAndReturn (ADR("dys santeam"));
      DerPron := "dys santeam";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 0.10 Frs ";
    IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ù
10 : DateStamp(DSR);
    NminEND°IndTemps§ := SHORT(DSR.dsMinute);
    NticksEND°IndTemps§ := SHORT(DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb005 := Nb005 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 0.05;
    IF Nb005 <= 7
    THEN
```

```
            Der.X := 170 + ((Nb005 - 1) * 18);
            Der.Y := 187;
            Der.Type := "PA";
            Der.Last := 0.05;
            AuMoinsUn := TRUE;
            Delay (25);
            DrawImage (Win^.RPort^,TabImages°10§,Der.X,Der.Y)
         ELSE
            AuMoinsUn := FALSE;
            Der.Last := 0.05
         END;
         IF (TableauParametres°1§ = 2) THEN
           Res := SayAndReturn (ADR("senk santeam"));
           DerPron := "senk santeam";
         END;
         wMove (Won^,7,1);
         wClrEndLine (Won^);
         ConvRealToString (Somme, StrSomme, 2, For);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,15,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);
         Trace := "Il appuie sur 0.05 Frs ";
         IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END ù
     11: IF ActivEff THEN
          IF (NminEND°IndTemps§ > NminBEG°IndTemps§) THEN
           NticksEND°IndTemps§ := NticksEND°IndTemps§ + (3000 * (NminEND°IndTem
s§ - NminBEG°IndTemps§));
          END;
          ent := (NticksEND°IndTemps§ - NticksBEG°IndTemps§);
          ree := real (ent);
          ree2 := (ree / 50.);
          TempsInterm°IndTemps§ := entier (ree2);
          IF IndTemps < 20 THEN
            IndTemps := IndTemps + 1;
          END;
           IF TableauChaine.Indice <= 499
           THEN IF NOT QR
                THEN TableauChaine.Tab°TableauChaine.Indice§ := 4;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                          = equal)
                     THEN TableauChaine.Tab °TableauChaine.Indice§ := 21;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
```

```
            ree := real(ent);
            TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
                  TableauChaine.Indice := TableauChaine.Indice + 1;
             ELSE TableauChaine.Tab °TableauChaine.Indice§ := 13;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
                  TableauChaine.Indice := TableauChaine.Indice + 1;
             END;
      END;
 END;
Trace := "Il demande d'effacer ";
IF TableauTraces.Indice <= 499 THEN
      CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
      TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet éè,GadgetActivationSet éEndGadget,
                 RelVerifyè);
AddGadgetTextButton (30,34, ADR("OUI"));
AddGadgetTextButton (115,34, ADR("NON"));
AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS EFFACER ?"));
GadgetOpt (GadgetFlagsSet éè, GadgetActivationSet éRelVerifyè,
           GadgetMutualExcludeSet éè);
GlEff := EndGadgetList ();
InitRequester (ReqEff);
WITH ReqEff DO
       OlderRequest := NIL;
       LeftEdge := 70;
       TopEdge := 8;
       Width := 180;
       Height := 50;
       ReqGadget := GlEff;
       ReqBorder := NIL;
       ReqText := NIL;
       Flags := RequesterFlagsSet éè;
       BackFill := BYTE(10);
       ReqLayer := NIL;
       ImageBMap := NIL;
END;
IF Request (ReqEff, Win^) THEN
   IF (TableauParametres°1§ = 2) THEN
     Res := SayAndReturn (ADR("Voolay voo effahsay ?"));
   END;
   DoneEff := FALSE;
   WHILE NOT DoneEff DO
       Sig := Wait(SignalSetéCARDINAL(Win^.UserPort^.mpSigBit)è);
       LOOP
          MsgEff := GetMsg (Win^.UserPort^);
          IF (MsgEff = NIL) THEN EXIT; END;
          ProcIMsg (WpEff, MsgEff);
       END;
   END;
END;
```

```
      FreeGadgetList (GlEff^);
      Delay(25);
      IF Efface THEN
        ActivEff := FALSE;
        IF ((CompareString (Der.Type,"BIL") = equal) AND (AuMoinsUn))
          THEN DrawImage (Win^.RPort^,TabImages°24§,Der.X,Der.Y);
            IF (Der.X # 170) THEN
              IF Der.Last = 100.
                THEN DrawImage (Win^.RPort^,TabImages°11§,Der.X - 20,Der.Y
                ELSIF Der.Last = 50.
                THEN DrawImage (Win^.RPort^,TabImages°12§,Der.X - 20,Der.Y
                ELSIF Der.Last = 20.
                THEN DrawImage (Win^.RPort^,TabImages°13§,Der.X - 20,Der.Y
                ELSIF Der.Last = 10.
                THEN DrawImage (Win^.RPort^,TabImages°14§,Der.X - 20,Der.Y
              END
            END
          ELSIF ((CompareString (Der.Type,"P5F") = equal) AND (AuMoinsUn))
            THEN DrawImage (Win^.RPort^,TabImages°25§,Der.X,Der.Y)
            ELSIF ((CompareString (Der.Type,"PA") = equal) AND (AuMoinsUn)
              THEN DrawImage (Win^.RPort^,TabImages°26§,Der.X,Der.Y)
        END;
        SommePrecedente := Somme;
        Somme := Somme - Der.Last;
        wMove (Won^,7,1);
        wClrEndLine (Won^);
        ConvRealToString (Somme, StrSomme, 2, For);
        PutStr (Won^,ADR(StrSomme));
        wMove (Won^,15,1);
        PutStr (Won^,ADR("Frs"));
        wSetCursor (Won^,FALSE);
        DerPron := "";
        IF Der.Last = 100.
         THEN  Nb100 := Nb100 - 1
         ELSIF  Der.Last = 50.
         THEN  Nb50 := Nb50 - 1
         ELSIF  Der.Last = 20.
         THEN  Nb20 := Nb20 - 1
         ELSIF  Der.Last = 10.
         THEN  Nb10 := Nb10 - 1
         ELSIF  Der.Last = 5.
         THEN  Nb5 := Nb5 - 1
         ELSIF  Der.Last = 2.
         THEN  Nb2 := Nb2 - 1
         ELSIF  Der.Last = 1.
         THEN  Nb1 := Nb1 - 1
         ELSIF  Der.Last = 0.5
         THEN  Nb05 := Nb05 - 1
         ELSIF  Der.Last = 0.2
         THEN  Nb02 := Nb02 - 1
         ELSIF  Der.Last = 0.1
         THEN  Nb01 := Nb01 - 1
         ELSIF Der.Last = 0.05
         THEN  Nb005 := Nb005 - 1
         ELSE
        END;
      END;
      DateStamp(DSR);
      NminBEG°IndTemps§ := SHORT(DSR.dsMinute);
      NminEND°IndTemps§ := NminBEG°IndTemps§;
      NticksBEG°IndTemps§ := SHORT(DSR.dsTick);
      NticksEND°IndTemps§ := NticksBEG°IndTemps§;
    END ù

12:   IF (NminEND°IndTemps§ > NminBEG°IndTemps§) THEN
       NticksEND°IndTemps§ := NticksEND°IndTemps§ +
```

```
    (3000 * (NminEND°IndTemps§ - NminBEG°IndTemps§));
END;
ent := (NticksEND°IndTemps§ - NticksBEG°IndTemps§);
ree := real (ent);
ree2 := (ree / 50.);
TempsInterm°IndTemps§ := entier (ree2);
IF IndTemps < 20 THEN
  IndTemps := IndTemps + 1;
END;
IF TableauChaine.Indice <= 499
THEN IF NOT QR
     THEN TableauChaine.Tab°TableauChaine.Indice§ := 6;
          DateStamp(DSR);
          ActMinute := SHORT(DSR.dsMinute);
          ActTick := SHORT(DSR.dsTick);
          ActTick2 := ActTick;
          IF ActMinute > AncMinute
          THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
          END;
          ent := ActTick - AncTick;
          ree := real(ent);
          TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
          AncMinute := ActMinute;
          AncTick := ActTick2;
          TableauChaine.Indice := TableauChaine.Indice + 1;
     ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
               = equal)
          THEN TableauChaine.Tab °TableauChaine.Indice§ := 23;
          DateStamp(DSR);
          ActMinute := SHORT(DSR.dsMinute);
          ActTick := SHORT(DSR.dsTick);
          ActTick2 := ActTick;
          IF ActMinute > AncMinute
          THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
          END;
          ent := ActTick - AncTick;
          ree := real(ent);
          TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
          AncMinute := ActMinute;
          AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          ELSE TableauChaine.Tab °TableauChaine.Indice§ := 15;
          DateStamp(DSR);
          ActMinute := SHORT(DSR.dsMinute);
          ActTick := SHORT(DSR.dsTick);
          ActTick2 := ActTick;
          IF ActMinute > AncMinute
          THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
          END;
          ent := ActTick - AncTick;
          ree := real(ent);
          TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
          AncMinute := ActMinute;
          AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
     END;
END;

Trace := "Il appuie sur OK ";
IF TableauTraces.Indice <= 499 THEN
       CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace)
       TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
```

```
        GlobalGadgetOpt (GadgetFlagsSet éè.GadgetActivationSet éEndGadget,
                         RelVerifyè);
        AddGadgetTextButton (30,34, ADR("OUI"));
        AddGadgetTextButton (115,34, ADR("NON"));
        AddGadgetTextButton (15,9, ADR("EST-CE BIEN JUSTE ?"));
        GadgetOpt (GadgetFlagsSet éè, GadgetActivationSet éRelVerifyè,
                   GadgetMutualExcludeSet éè);
        GlFin := EndGadgetList ();
        InitRequester (ReqFin);
        WITH ReqFin DO
              OlderRequest := NIL;
              LeftEdge := 70;
              TopEdge := 8;
              Width := 180;
              Height := 50;
              ReqGadget := GlFin;
              ReqBorder := NIL;
              ReqText := NIL;
              Flags := RequesterFlagsSet éè;
              BackFill := BYTE(10);
              ReqLayer := NIL;
              ImageBMap := NIL;
        END;
        IF Request (ReqFin, Win^) THEN
           IF (TableauParametres°1§ = 2) THEN
               Res := SayAndReturn (ADR("ahvay voo feeny ?"));
           END;
           DoneFin := FALSE;
           WHILE NOT DoneFin DO
               Sig := Wait(SignalSetéCARDINAL(Win^.UserPort^.mpSigBit)è);
               LOOP
                  MsgFin := GetMsg (Win^.UserPort^);
                  IF (MsgFin = NIL) THEN EXIT; END;
                  ProcIMsg (WpFin, MsgFin);
               END;
           END;
        END;
        FreeGadgetList (GlFin^);
        Delay(25);
        IF Fini THEN Done := TRUE
        ELSE
          DateStamp(DSR);
          NminBEG°IndTemps§ := SHORT(DSR.dsMinute);
          NminEND°IndTemps§ := NminBEG°IndTemps§;
          NticksBEG°IndTemps§ := SHORT(DSR.dsTick);
          NticksEND°IndTemps§ := NticksBEG°IndTemps§;

        ENDù
13 :  IF QR = TRUE
      THEN
        IF (NminEND°IndTemps§ > NminBEG°IndTemps§) THEN
         NticksEND°IndTemps§ := NticksEND°IndTemps§ +
                        (3000 * (NminEND°IndTemps§ - NminBEG°IndTemps§));
        END;
        ent := (NticksEND°IndTemps§ - NticksBEG°IndTemps§);
        ree := real (ent);
        ree2 := (ree / 50.);
        TempsInterm°IndTemps§ := entier (ree2);
        IF IndTemps < 20 THEN
          IndTemps := IndTemps + 1;
        END;

         IF TableauChaine.Indice <= 499
         THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                           = equal)
              THEN TableauChaine.Tab °TableauChaine.Indice§ := 22;
```

```
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
        ELSE TableauChaine.Tab °TableauChaine.Indice§ := 14;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
        END;
    END;
    Trace := "Il appuie sur QUITTER ";
    IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END;
    BeginGadgetList ();
    GadgetTypeReq := TRUE;
    GlobalGadgetOpt (GadgetFlagsSet éè,GadgetActivationSet éEndGadget,
                     RelVerifyè);
    AddGadgetTextButton (30,34, ADR("OUI"));
    AddGadgetTextButton (115,34, ADR("NON"));
    AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS QUITTER ?"));
    GadgetOpt (GadgetFlagsSet éè, GadgetActivationSet éRelVerifyè,
            GadgetMutualExcludeSet éè);
    GlQui := EndGadgetList ();
    InitRequester (ReqQui);
    WITH ReqQui DO
        OlderRequest := NIL;
        LeftEdge := 70;
        TopEdge := 8;
        Width := 180;
        Height := 50;
        ReqGadget := GlQui;
        ReqBorder := NIL;
        ReqText := NIL;
        Flags := RequesterFlagsSet éè;
        BackFill := BYTE(10);
        ReqLayer := NIL;
        ImageBMap := NIL;
    END;
    IF Request (ReqQui, Win^) THEN
       IF (TableauParametres°1§ = 2) THEN
         Res := SayAndReturn (ADR("Voolay voo kittay ?"));
       END;
       DoneQui := FALSE;
       WHILE NOT DoneQui DO
         Sig := Wait(SignalSetéCARDINAL(Win^.UserPort^.mpSigBit)è);
```

```
        LOOP
           MsgQui := GetMsg (Win^.UserPort^);
           IF (MsgQui = NIL) THEN EXIT; END;
           ProcIMsg (WpQui, MsgQui);
        END;
     END;
    END;
    FreeGadgetList (GlQui^);
    Delay(25);
    IF Quitter THEN Done := TRUE;ES := "quitter";
    ELSE

    DateStamp(DSR);
    NminBEG°IndTemps§ := SHORT(DSR.dsMinute);
    NminEND°IndTemps§ := NminBEG°IndTemps§;
    NticksBEG°IndTemps§ := SHORT(DSR.dsTick);
    NticksEND°IndTemps§ := NticksBEG°IndTemps§;
  END;
ELSE
  IF (NminEND°IndTemps§ > NminBEG°IndTemps§) THEN
   NticksEND°IndTemps§ := NticksEND°IndTemps§ +
                        (3000 * (NminEND°IndTemps§ - NminBEG°IndTemps§));
  END;
  ent := (NticksEND°IndTemps§ - NticksBEG°IndTemps§);
  ree := real (ent);
  ree2 := (ree / 50.);
  TempsInterm°IndTemps§ := entier (ree2);
  IF IndTemps < 20 THEN
    IndTemps := IndTemps + 1;
  END;
   IF TableauChaine.Indice <= 499
   THEN TableauChaine.Tab°TableauChaine.Indice§ := 5;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps °TableauChaine.Indice§ := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
       TableauChaine.Indice := TableauChaine.Indice + 1;
   END;
   Trace := "Il appuie sur RECOMMENCER ";
   IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab °TableauTraces.Indice§ ,Trace);
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END;
   BeginGadgetList ();
   GadgetTypeReq := TRUE;
   GlobalGadgetOpt (GadgetFlagsSet éè,GadgetActivationSet éEndGadget,
                    RelVerifyè);
   AddGadgetTextButton (30,34, ADR("OUI"));
   AddGadgetTextButton (115,34, ADR("NON"));
   AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS RECOMMENCER ?"));
   GadgetOpt (GadgetFlagsSet éè, GadgetActivationSet éRelVerifyè,
            GadgetMutualExcludeSet éè);
   GlRec := EndGadgetList ();
   InitRequester (ReqRec);
   WITH ReqRec DO
        OlderRequest := NIL;
        LeftEdge := 50;
        TopEdge := 8;
```

```
      Width := 220;
      Height := 50;
      ReqGadget := GlRec;
      ReqBorder := NIL;
      ReqText := NIL;
      Flags := RequesterFlagsSet éè;
      BackFill := BYTE(10);
      ReqLayer := NIL;
      ImageBMap := NIL;
END;
IF Request (ReqRec, Win^) THEN
   IF (TableauParametres°1§ = 2) THEN
     Res := SayAndReturn (ADR("Vooley voo ra commohsay ?"));
   END;
   DoneRec := FALSE;
   WHILE NOT DoneRec DO
     Sig := Wait(SignalSetéCARDINAL(Win^.UserPort^.mpSigBit)è);
     LOOP
        MsgRec := GetMsg (Win^.UserPort^);
        IF (MsgRec = NIL) THEN EXIT; END;
        ProcIMsg (WpRec, MsgRec);
     END;
 END;
END;
FreeGadgetList (GlRec^);
Delay(25);
IF Recom THEN
      DrawImage (Win^.RPort^,TabImages°24§,170,17);
      DrawImage (Win^.RPort^,TabImages°24§,190,17);
      DrawImage (Win^.RPort^,TabImages°24§,210,17);
      DrawImage (Win^.RPort^,TabImages°24§,230,17);
      DrawImage (Win^.RPort^,TabImages°24§,250,17);
      DrawImage (Win^.RPort^,TabImages°24§,170,34);
      DrawImage (Win^.RPort^,TabImages°24§,190,34);
      DrawImage (Win^.RPort^,TabImages°24§,210,34);
      DrawImage (Win^.RPort^,TabImages°24§,230,34);
      DrawImage (Win^.RPort^,TabImages°24§,250,34);
      DrawImage (Win^.RPort^,TabImages°24§,170,51);
      DrawImage (Win^.RPort^,TabImages°24§,190,51);
      DrawImage (Win^.RPort^,TabImages°24§,210,51);
      DrawImage (Win^.RPort^,TabImages°24§,230,51);
      DrawImage (Win^.RPort^,TabImages°24§,250,51);
      DrawImage (Win^.RPort^,TabImages°24§,170,68);
      DrawImage (Win^.RPort^,TabImages°24§,190,68);
      DrawImage (Win^.RPort^,TabImages°24§,210,68);
      DrawImage (Win^.RPort^,TabImages°24§,230,68);
      DrawImage (Win^.RPort^,TabImages°24§,250,68);
      DrawImage (Win^.RPort^,TabImages°25§,170,84);
      DrawImage (Win^.RPort^,TabImages°25§,194,84);
      DrawImage (Win^.RPort^,TabImages°25§,218,84);
      DrawImage (Win^.RPort^,TabImages°25§,242,84);
      DrawImage (Win^.RPort^,TabImages°25§,266,84);
      DrawImage (Win^.RPort^,TabImages°26§,170,107);
      DrawImage (Win^.RPort^,TabImages°26§,188,107);
      DrawImage (Win^.RPort^,TabImages°26§,206,107);
      DrawImage (Win^.RPort^,TabImages°26§,224,107);
      DrawImage (Win^.RPort^,TabImages°26§,242,107);
      DrawImage (Win^.RPort^,TabImages°26§,260,107);
      DrawImage (Win^.RPort^,TabImages°26§,278,107);
      DrawImage (Win^.RPort^,TabImages°26§,170,125);
      DrawImage (Win^.RPort^,TabImages°26§,188,125);
      DrawImage (Win^.RPort^,TabImages°26§,206,125);
      DrawImage (Win^.RPort^,TabImages°26§,224,125);
      DrawImage (Win^.RPort^,TabImages°26§,242,125);
      DrawImage (Win^.RPort^,TabImages°26§,260,125);
      DrawImage (Win^.RPort^,TabImages°26§,278,125);
```

```
            DrawImage (Win^.RPort^,TabImages°26§,170,143);
            DrawImage (Win^.RPort^,TabImages°26§,188,143);
            DrawImage (Win^.RPort^,TabImages°26§,206,143);
            DrawImage (Win^.RPort^,TabImages°26§,224,143);
            DrawImage (Win^.RPort^,TabImages°26§,242,143);
            DrawImage (Win^.RPort^,TabImages°26§,260,143);
            DrawImage (Win^.RPort^,TabImages°26§,278,143);
            DrawImage (Win^.RPort^,TabImages°26§,170,161);
            DrawImage (Win^.RPort^,TabImages°26§,188,161);
            DrawImage (Win^.RPort^,TabImages°26§,206,161);
            DrawImage (Win^.RPort^,TabImages°26§,224,161);
            DrawImage (Win^.RPort^,TabImages°26§,242,161);
            DrawImage (Win^.RPort^,TabImages°26§,260,161);
            DrawImage (Win^.RPort^,TabImages°26§,278,161);
            DrawImage (Win^.RPort^,TabImages°26§,170,179);
            DrawImage (Win^.RPort^,TabImages°26§,188,179);
            DrawImage (Win^.RPort^,TabImages°26§,206,179);
            DrawImage (Win^.RPort^,TabImages°26§,224,179);
            DrawImage (Win^.RPort^,TabImages°26§,242,179);
            DrawImage (Win^.RPort^,TabImages°26§,260,179);
            DrawImage (Win^.RPort^,TabImages°26§,278,179);
            DrawImage (Win^.RPort^,TabImages°26§,170,197);
            DrawImage (Win^.RPort^,TabImages°26§,188,197);
            DrawImage (Win^.RPort^,TabImages°26§,206,197);
            DrawImage (Win^.RPort^,TabImages°26§,224,197);
            DrawImage (Win^.RPort^,TabImages°26§,242,197);
            DrawImage (Win^.RPort^,TabImages°26§,260,197);
            DrawImage (Win^.RPort^,TabImages°26§,278,197);

            Somme := 0.;
            wMove (Won^,7,1);
            wClrEndLine (Won^);
            ConvRealToString (Somme, StrSomme, 2, For);
            PutStr (Won^,ADR(StrSomme));
            wMove (Won^,15,1);
            PutStr (Won^,ADR("Frs"));
            wSetCursor (Won^,FALSE);
            ES := "";
            SommeMax := Max;
            Nb100 := 0;
            Nb50 := 0;
            Nb20 := 0;
            Nb10 := 0;
            Nb5 := 0;
            Nb2 := 0;
            Nb1 := 0;
            Nb05 := 0;
            Nb02 := 0;
            Nb01 := 0;
            Nb005 := 0;
            Der.Last := 0.;
            Der.X := 150;
            AuMoinsUn := FALSE;
            ActivEff := FALSE;
            For := Decimal;
            CopyString (DerPron,PPron);
        END;
      DateStamp(DSR);
      NminBEG°IndTemps§ := SHORT(DSR.dsMinute);
      NminEND°IndTemps§ := NminBEG°IndTemps§;
      NticksBEG°IndTemps§ := SHORT(DSR.dsTick);
      NticksEND°IndTemps§ := NticksBEG°IndTemps§;
       ENDù
14:   IF (TableauParametres°1§ = 2) THEN
         Res := SayAndReturn (ADR(DerPron));
      END;
```

```
  END;
CloseSpeech ();
END;
END GadgetHandlerBS;


PROCEDURE SaisieSommeBillets (VAR Phrase : ARRAY OF CHAR;
                              VAR PhrasePron : ARRAY OF CHAR;
                              Quit :BOOLEAN; VAR Valeur : REAL;
                              VAR EtatSortie : Str20);



BEGIN
    CopyString (PPron,PhrasePron);
    ES := "";
    QR := Quit;
    Somme := 0.;
    SommeMax := Max;
    Nb100 := 0;
    Nb50 := 0;
    Nb20 := 0;
    Nb10 := 0;
    Nb5 := 0;
    Nb2 := 0;
    Nb1 := 0;
    Nb05 := 0;
    Nb02 := 0;
    Nb01 := 0;
    Nb005 := 0;
    Der.Last := 0.;
    Der.X := 150;
    AuMoinsUn := FALSE;
    ActivEff := FALSE;
    For := Decimal;
    Nbillets := 0;
    IndTemps := 1;

    WITH Wp DO
      procGadgetUp := GadgetHandlerBS;
    END;
    WITH WpEff DO
      procGadgetUp := GadgetHandlerEffBS;
    END;
    WITH WpQui DO
      procGadgetUp := GadgetHandlerQuiBS;
    END;
    WITH WpFin DO
      procGadgetUp := GadgetHandlerFinBS;
    END;
    WITH WpRec DO
      procGadgetUp := GadgetHandlerRecBS;
    END;
    Scr := CreateScreen (320,230,5,NIL);
    IF (Scr # NIL) THEN
      cmap^[00] := 0000H;
      cmap^[01] := 0FFFH;
      cmap^[02] := 0E00H;
      cmap^[03] := 0A00H;
      cmap^[04] := 0D80H;
      cmap^[05] := 0FE0H;
      cmap^[06] := 0FCAH;
      cmap^[07] := 0080H;
      cmap^[08] := 00B6H;
      cmap^[09] := 00DDH;
```

```
cmap°10§ := 00AFH;
cmap°11§ := 007CH;
cmap°12§ := 000FH;
cmap°13§ := 070FH;
cmap°14§ := 0C0EH;
cmap°15§ := 0C08H;
cmap°16§ := 0620H;
cmap°17§ := 0E52H;
cmap°18§ := 0A52H;
cmap°19§ := 0FCAH;
cmap°20§ := 0333H;
cmap°21§ := 0444H;
cmap°22§ := 0555H;
cmap°23§ := 0666H;
cmap°24§ := 0777H;
cmap°25§ := 0888H;
cmap°26§ := 0999H;
cmap°27§ := 0AAAH;
cmap°28§ := 0CCCH;
cmap°29§ := 0DDDH;
cmap°30§ := 0EEEH;
cmap°31§ := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

ImgPtr := FindImageTable(33333333H, ImgCount);
IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
  Indice := 0;
  WHILE (Indice <= ImgCount-1) DO
    WITH TabImages°Indice§ DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr^°Indice§.Width;
      Height      := ImgPtr^°Indice§.Height;
      Depth       := ImgPtr^°Indice§.Depth;
      ImageData   := ImgPtr^°Indice§.Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

ImgPtr2 := FindImageTable(44444444H, ImgCount2);
IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
 WHILE (Indice-ImgCount <= ImgCount2-1) DO
    WITH TabImages°Indice§ DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr2^°Indice-ImgCount§.Width;
      Height      := ImgPtr2^°Indice-ImgCount§.Height;
      Depth       := ImgPtr2^°Indice-ImgCount§.Depth;
      ImageData   := ImgPtr2^°Indice-ImgCount§.Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;
  ImgPtr3 := FindImageTable(77777777H, ImgCount3);
  IF (ImgPtr3 # NIL) AND (ImgCount3 # 0) THEN
    WITH TabImages°Indice§ DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr3^°9§.Width;
      Height      := ImgPtr3^°9§.Height;
      Depth       := ImgPtr3^°9§.Depth;
```

```
        ImageData   := ImgPtr3^°3§.Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
     END;
     Indice := Indice + 1;

 ImgPtr4 := FindImageTable(66666666H, ImgCount4);
 IF (ImgPtr4 # NIL) AND (ImgCount4 # 0) THEN
   list6 := 3;
   WHILE (list6 <= ImgCount4-1) DO
     WITH TabImages°Indice§ DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr4^°list6§.Width;
       Height      := ImgPtr4^°list6§.Height;
       Depth       := ImgPtr4^°list6§.Depth;
       ImageData   := ImgPtr4^°list6§.Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list6 := list6 + 1;
     Indice := Indice + 1;
   END;
ImgPtr5 := FindImageTable(11111114H, ImgCount5);
 IF (ImgPtr5 # NIL) AND (ImgCount5 # 0) THEN
  list7 := 3;
  WHILE (list7 <= ImgCount5-1) DO
     WITH TabImages°Indice§ DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr5^°list7§.Width;
       Height      := ImgPtr5^°list7§.Height;
       Depth       := ImgPtr5^°list7§.Depth;
       ImageData   := ImgPtr5^°list7§.Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list7 := list7 + 1;
     Indice := Indice + 1;
   END;


    BeginGadgetList();
    AddGadgetImageButton (10,10,TabImages°0§);
    AddGadgetImageButton (10,47,TabImages°1§);
    AddGadgetImageButton (10,84,TabImages°2§);
    AddGadgetImageButton (10,121,TabImages°3§);
    AddGadgetImageButton (80,5,TabImages°4§);
    AddGadgetImageButton (85,45,TabImages°5§);
    AddGadgetImageButton (87,78,TabImages°6§);
    AddGadgetImageButton (90,109,TabImages°7§);
    AddGadgetImageButton (90,132,TabImages°8§);
    AddGadgetImageButton (91,156,TabImages°9§);
    AddGadgetImageButton (92,178,TabImages°10§);
    AddGadgetImageButton (4,200,TabImages°20§);
    AddGadgetImageButton (121,200,TabImages°21§);
    IF Quit = TRUE THEN AddGadgetImageButton (63,200,TabImages°22§)
                   ELSE AddGadgetImageButton (63,200,TabImages°36§) END;
          IF TableauParametres°1§ = 2 THEN
    AddGadgetImageButton (6,160,TabImages°35§);
    END;
    G1 := EndGadgetList();
```

```
IF (Gl # NIL) THEN
   Win := CreateWindow (0,0,320,230,WIDCMP,WFlags,Gl,Scr,NIL);

   Win2 := CreateWindow (134,72,23,102,WIDCMP,WFlags2,NIL,Scr,NIL);

   Won := CreateWindow (168,212,152,15,WIDCMP,WFlags,NIL,Scr,NIL);
   IF (Win # NIL) AND (Won # NIL)  THEN

   IF CreateConsole (Won^) THEN
      wClrScr (Won^);
      PutStr (Won^,ADR("Total"));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      DrawImage (Win^.RPort^,TabImages°28§,165,0);
      DrawImage (Win^.RPort^,TabImages°28§,165,100);
      IF NOT QR THEN
        DrawImage (Win^.RPort^,TabImages°43§,0,0);
        DrawImage (Win^.RPort^,TabImages°43§,130,0);
        DrawImage (Win^.RPort^,TabImages°43§,190,0);
        DrawImage (Win^.RPort^,TabImages°43§,0,227);
        DrawImage (Win^.RPort^,TabImages°43§,130,227);
        DrawImage (Win^.RPort^,TabImages°43§,190,227);
        DrawImage (Win^.RPort^,TabImages°45§,0,0);
        DrawImage (Win^.RPort^,TabImages°45§,0,100);
        DrawImage (Win^.RPort^,TabImages°45§,317,0);
        DrawImage (Win^.RPort^,TabImages°45§,317,100);
        DrawImage (Win^.RPort^,TabImages°42§,123,3);
        DrawImage (Win^.RPort^,TabImages°45§,165,0);
        DrawImage (Win^.RPort^,TabImages°45§,165,100);
        DrawImage (Win^.RPort^,TabImages°39§,175,5);

      END;
      IF (CompareString (PhrasePron,"Cobya ah vay voo ra su darjo ?")
              = equal)
      THEN
        DrawImage (Win^.RPort^,TabImages°31§,0,0);
        DrawImage (Win^.RPort^,TabImages°31§,130,0);
        DrawImage (Win^.RPort^,TabImages°31§,190,0);
        DrawImage (Win^.RPort^,TabImages°31§,0,227);
        DrawImage (Win^.RPort^,TabImages°31§,130,227);
        DrawImage (Win^.RPort^,TabImages°31§,190,227);
        DrawImage (Win^.RPort^,TabImages°29§,0,0);
        DrawImage (Win^.RPort^,TabImages°29§,0,100);
        DrawImage (Win^.RPort^,TabImages°29§,317,0);
        DrawImage (Win^.RPort^,TabImages°29§,317,100);
        DrawImage (Win^.RPort^,TabImages°33§,123,3);
        DrawImage (Win^.RPort^,TabImages°29§,165,0);
        DrawImage (Win^.RPort^,TabImages°29§,165,100);
        DrawImage (Win^.RPort^,TabImages°41§,175,5);

      ELSIF (CompareString (PhrasePron,"Cobya ah vay voo daypensay ?"
              = equal)
      THEN
        DrawImage (Win^.RPort^,TabImages°32§,0,0);
        DrawImage (Win^.RPort^,TabImages°32§,130,0);
        DrawImage (Win^.RPort^,TabImages°32§,190,0);
        DrawImage (Win^.RPort^,TabImages°32§,0,227);
        DrawImage (Win^.RPort^,TabImages°32§,130,227);
        DrawImage (Win^.RPort^,TabImages°32§,190,227);
        DrawImage (Win^.RPort^,TabImages°30§,0,0);
        DrawImage (Win^.RPort^,TabImages°30§,0,100);
        DrawImage (Win^.RPort^,TabImages°30§,317,0);
        DrawImage (Win^.RPort^,TabImages°30§,317,100);
        DrawImage (Win^.RPort^,TabImages°34§,123,3);
        DrawImage (Win^.RPort^,TabImages°30§,165,0);
```

```
                DrawImage (Win^.RPort^,TabImages°30§,165,100);
                DrawImage (Win^.RPort^,TabImages°40§,175,5);

              END;
              IF TableauParametres°1§ = 2 THEN
                IF OpenSpeech () THEN
                      Res := SayAndReturn (ADR(PhrasePron));
                      CopyString (DerPron,PhrasePron);
                      CloseSpeech ()
                END;
              END;

              SA := Sactuel;
              DateStamp (DSR);
              NminBEG°IndTemps§ := SHORT (DSR.dsMinute);
              NticksBEG°IndTemps§ := SHORT (DSR.dsTick);
              Done := FALSE;
              WHILE (NOT Done) DO
                Sig := Wait(SignalSetéCARDINAL(Win^.UserPort^.mpSigBit)è);
                LOOP
                  Msg := GetMsg(Win^.UserPort^);
                  IF (Msg = NIL) THEN EXIT; END;
                  ProcIMsg (Wp, Msg);
                END;
              END;
              IndTemps := IndTemps - 1;
              IF (( Nbillets > 0) AND (NOT QR) )
              THEN
                 II := 1;
                 WHILE (II <= IndTemps) DO
                    TempsSIBillets := TempsSIBillets + TempsInterm°II§;
                    II := II + 1;
                 END;
                 ree := real(TempsSIBillets);
                 ree2 := real (Nbillets);
                 TempsSIBillets := entier (ree / ree2);
              END;
              DeleteConsole (Won^);
              ELSE WriteString ("Console non créée")
            END;
          EtatSortie := ES;
          Valeur := Somme;

          CloseWindow(Won^);
          CloseWindow(Win^);
          END;
          FreeGadgetList (Gl^);
        END;
      CloseScreen(Scr^);
    END;
   END;
  END;
 END;
END;
IF Quit = TRUE
 THEN
  TabNombre[0] := Nb100;
  TabNombre[1] := Nb50;
  TabNombre[2] := Nb20;
  TabNombre[3] := Nb10;
  TabNombre[4] := Nb5;
  TabNombre[5] := Nb2;
  TabNombre[6] := Nb1;
  TabNombre[7] := Nb05;
  TabNombre[8] := Nb02;
  TabNombre[9] := Nb01;
```

```
    TabNombre[10] := Nb005;
  ELSE
    TabNombreInit[0] := Nb100;
    TabNombreInit[1] := Nb50;
    TabNombreInit[2] := Nb20;
    TabNombreInit[3] := Nb10;
    TabNombreInit[4] := Nb5;
    TabNombreInit[5] := Nb2;
    TabNombreInit[6] := Nb1;
    TabNombreInit[7] := Nb05;
    TabNombreInit[8] := Nb02;
    TabNombreInit[9] := Nb01;
    TabNombreInit[10] := Nb005;
END;

END SaisieSommeBillets;

(*  VAR
    p,pp : ARRAY [0..40] OF CHAR;
    q : BOOLEAN;
    v : REAL;
    e : Str20;

BEGIN

    p := "Combien avez-vous d'argent ?";
    pp := "Cobya ah vay voo darjo ?";
    q := FALSE;
    SaisieSommeBillets (p,pp,q,v,e);  *)

END SSBillets.
```

```
IMPLEMENTATION MODULE SRBillets;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request,RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100,TabNombreInit,
                              TableauTraces,TRecettes,TDepenses,
                              TableauParametres,Sinit,Max,TabNombre,Sactuel;
FROM Utilitaires IMPORT ConvPrononcable,RegletteJour,RegHeure;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                              ConvStringToReal;


CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;

VAR
  Done,DoneOK : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  Gl,GlEff,GlQui,GlFin,GlOK : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
```

```
  Confirmation,Trace : Str40;
  Der : Dernier;
  Somme, SommePrecedente, SommeMax,ValeurMax,ValeurPrec : REAL;
  StrSomme,ES,fonc,SommeString,Maxaf : Str20;
  TePtr : IntuiTextPtr;
  For : RealToStringFormat;
  Win,Won,Win1,Win2,Win3,Win4,Win5,Win6,Win61,Wun,Wun2 : WindowPtr;
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  TabImages : ARRAY [0..29] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice,i,I,X,Y : CARDINAL;
  Res : BOOLEAN;
  SomPron,PhrasePron : Str100;
  Phrase  : Str40;
  Phrase2,Diff : ARRAY [0..39] OF CHAR;


PROCEDURE GadgetHandler (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN

  Done := TRUE;
END GadgetHandler;

PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : DoneOK := TRUE |
     1 :
  END;
END GadgetHandlerOK;

PROCEDURE OKRequester(EPMReq : INTEGER);
BEGIN


  WITH WpOK DO
      procGadgetUp := GadgetHandlerOK;
  END;

          IF OpenSpeech () THEN
          BeginGadgetList ();
          GadgetTypeReq := TRUE;
          GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                           RelVerify});
          AddGadgetTextButton (125,40, ADR(" OK "));
          IF (EPMReq = 1)
           THEN ConvRealToString (Sinit - Sactuel,Diff,2,Decimal);
                CopyString (Phrase2,"Il y a une erreur de ");
                ConcatString (Phrase2,Diff);
                ConcatString (Phrase2," Frs.")
           ELSE ConvRealToString (Sactuel - Sinit,Diff,2,Decimal);
                CopyString (Phrase2,"Il y a une erreur de ");
                ConcatString (Phrase2,Diff);
                ConcatString (Phrase2," Frs.")

          END;
          AddGadgetTextButton (2,15, ADR(Phrase2));
          GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                     GadgetMutualExcludeSet {});
          GlOK := EndGadgetList ();
          InitRequester (ReqOK);
          WITH ReqOK DO
                 OlderRequest := NIL;
                 LeftEdge := 3;
                 TopEdge := 65;
```

```
                Width := 268;
                Height := 60;
                ReqGadget := GlOK;
                ReqBorder := NIL;
                ReqText := NIL;
                Flags := RequesterFlagsSet {};
                BackFill := BYTE(10);
                ReqLayer := NIL;
                ImageBMap := NIL;
            END;
            IF Request (ReqOK, Win1^) THEN
            IF (EPMReq = 1) THEN
              IF (TableauParametres[1] = 2)
                THEN Res := SayAndReturn (ADR("Il y a une erreur"));
              END;
            ELSE
              IF (TableauParametres[1] = 2)
                THEN Res := SayAndReturn (ADR("Il y a une erreur"));
              END;
            END;
            DoneOK := FALSE;
            WHILE NOT DoneOK DO
              Sig := Wait(SignalSet{CARDINAL(Win1^.UserPort^.mpSigBit)});
              LOOP
                MsgOK := GetMsg (Win1^.UserPort^);
                IF (MsgOK = NIL) THEN EXIT; END;
                ProcIMsg (WpOK, MsgOK);
              END;
            END;
            END;
             FreeGadgetList (GlOK^);
CloseSpeech();
END;
END OKRequester;



PROCEDURE SommeRestanteBillets (EPMReq : INTEGER);

BEGIN

 IF OpenSpeech () THEN END;
  ValeurMax := Max;

  WITH Wp DO
      procGadgetUp := GadgetHandler;
  END;
  Scr := CreateScreen (320,240,5,NIL);
  IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
```

```
cmap[16] := 0E00H;
cmap[17] := 0E52H;
cmap[18] := 0A52H;
cmap[19] := 0FCAH;
cmap[20] := 0333H;
cmap[21] := 0444H;
cmap[22] := 0555H;
cmap[23] := 0666H;
cmap[24] := 0777H;
cmap[25] := 0888H;
cmap[26] := 0999H;
cmap[27] := 0AAAH;
cmap[28] := 0CCCH;
cmap[29] := 0DDDH;
cmap[30] := 0E00H;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

ImgPtr := FindImageTable(33333333H, ImgCount);
  Indice := 0;
  WHILE (Indice <= 2) DO
    WITH TabImages[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr^[Indice+11].Width;
      Height      := ImgPtr^[Indice+11].Height;
      Depth       := ImgPtr^[Indice+11].Depth;
      ImageData   := ImgPtr^[Indice+11].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

ImgPtr2 := FindImageTable(44444444H, ImgCount2);
  WHILE (Indice <= 6) DO
    WITH TabImages[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr2^[Indice-3].Width;
      Height      := ImgPtr2^[Indice-3].Height;
      Depth       := ImgPtr2^[Indice-3].Depth;
      ImageData   := ImgPtr2^[Indice-3].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;

WITH TabImages[7] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr^[7].Width;
      Height      := ImgPtr^[7].Height;
      Depth       := ImgPtr^[7].Depth;
      ImageData   := ImgPtr^[7].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
END;

WITH TabImages[8] DO
      LeftEdge    := 0;
      TopEdge     := 0;
```

```
        Width       := ImgPtr2^[4].Width;
        Height      := ImgPtr2^[4].Height;
        Depth       := ImgPtr2^[4].Depth;
        ImageData   := ImgPtr2^[4].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;

WITH TabImages[9] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr2^[5].Width;
        Height      := ImgPtr2^[5].Height;
        Depth       := ImgPtr2^[5].Depth;
        ImageData   := ImgPtr2^[5].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;

WITH TabImages[10] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr^[10].Width;
        Height      := ImgPtr^[10].Height;
        Depth       := ImgPtr^[10].Depth;
        ImageData   := ImgPtr^[10].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;

BeginGadgetList();
AddGadgetTextButton (1,4,ADR (" OK "));
Gl := EndGadgetList ();
IF (Gl # NIL) THEN
Win  := CreateWindow (0,0,320,240,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win # NIL) THEN
Win5 := CreateWindow (0,212,320,28,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win5 # NIL) THEN
Win4 := CreateWindow (0,195,275,17,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win4 # NIL) THEN
Win2 := CreateWindow (275,0,45,195,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win2 # NIL) THEN
Win6 := CreateWindow (285,40,24,102,WIDCMP,WFlags2,NIL,Scr,NIL);

IF (Win6 # NIL)  THEN
Win1 := CreateWindow (0,0,275,195,WIDCMP,WFlags,NIL,Scr,NIL);
IF (Win1 # NIL) THEN

Win3 := CreateWindow (275,195,45,17,WIDCMP,WFlags2,Gl,Scr,NIL);
IF (Win3 # NIL) THEN


i := 0;
X := 5;
WHILE (i <= 1) DO
  I := 1;
  Y := 75;
  WHILE (I <= TabNombreInit[i]) AND (I <= 4) DO
    DrawImage (Win1^.RPort^,TabImages[i],X,Y);
    Y := Y - 20;
    I := I + 1;
  END;
  X := X + 40;
```

```
    i := i + 1;
  END;

  i := 2;
  X := 5;
  WHILE (i <= 3) DO
    I := 1;
    Y := 170;
    WHILE (I <= TabNombreInit[i]) AND (I <= 4) DO
      DrawImage (Win1^.RPort^,TabImages[i],X,Y);
      Y := Y - 20;
      I := I + 1;
    END;
    X := X + 40;
    i := i + 1;
  END;

  I := 1;
  X := 90;
  Y := 168;
  WHILE (I <= TabNombreInit[4]) AND (I <= 7) DO
      DrawImage (Win1^.RPort^,TabImages[4],X,Y);
      Y := Y - 25;
      I := I + 1;
    END;

  i := 5;
  X := 130;
  WHILE (i <= 10) DO
    I := 1;
    Y := 170;
    WHILE (I <= TabNombreInit[i]) AND (I <= 9) DO
      DrawImage (Win1^.RPort^,TabImages[i],X,Y);
      Y := Y - 20;
      I := I + 1;
    END;
    X := X + 25;
    i := i + 1;
  END;

  ConvRealToString (Sinit,SommeString,2,Decimal);
  IF (TableauParametres[4] = 3) OR (TableauParametres[4] = 4)
                                OR (TableauParametres[4] = 6)
  THEN
    IF CreateConsole (Win4^)
      THEN
       CopyString (Phrase,"Il vous reste : ");
       ConcatString (Phrase,SommeString);
       ConcatString (Phrase," Francs");
       wSetCursor (Win4^,FALSE);
       wMove (Win4^,1,2);
       PutStr (Win4^,ADR(Phrase));
       DeleteConsole (Win4^);
    END;
END;
    IF (TableauParametres[4] = 4) OR (TableauParametres[4] = 5) OR
       (TableauParametres[4] = 6) THEN
      CopyString (PhrasePron,"il voo rest ");
      ConvPrononcable (SommeString,SomPron);
      ConcatString (PhrasePron,SomPron);
      Res := SayAndReturn (ADR(PhrasePron));

    END;
    CloseSpeech();

  ValeurPrec := 0.;
```

```
      Done := FALSE;
      WHILE (NOT Done) DO
        Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
        LOOP
          Msg := GetMsg(Win3^.UserPort^);
          IF (Msg = NIL) THEN EXIT;  END;
          ProcIMsg (Wp, Msg);
        END;
      END;
      CloseWindow(Win3^);
      END;
      CloseWindow(Win1^);
      END;
      CloseWindow (Win6^);
      END;
      CloseWindow(Win2^);
      END;
      CloseWindow(Win4^);
      END;
      CloseWindow(Win5^);
      END;
      CloseWindow(Win^);
      END;
      FreeGadgetList (Gl^);
      END;
      CloseScreen(Scr^);
      END;
END SommeRestanteBillets;

(*  VAR u: INTEGER;

BEGIN
   u := 1;
   SommeRestanteBillets (u); *)

END SRBillets.
```

Le programme Comptes : les modules spécifiques à la version française

```
IMPLEMENTATION MODULE CorrPMBillets;

FROM SYSTEM IMPORT SHORT;BYTE, ADR;
FROM AmigaDOS IMPORT DateStamp, DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText, IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton,AddGadgetImageButton, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20, Str40,Str100,TabNombreInit,
                              TableauTraces,TRecettes,TDepenses,
                              TableauParametres,Sinit,Max,TabNombre,Sactuel,
                              TableauChaine,AncMinute,AncTick,TableauTemps ;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                              ConvStringToReal;
FROM Utilitaires IMPORT ConvPrononcable;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;
VAR
  Done,DoneOK,DoneRec,Recom : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff,QR : BOOLEAN;
  Gl,GlEff,GlQui,GlFin,GlOK,GlRec : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK,WpRec : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK,ReqRec : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK,MsgRec : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb200,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  Trace : Str40;
  Der : Dernier;
  Somme,SommePrecedente, SommeMax,ValeurMax,ValeurPrec,SomAct,SA,SAP,UUU : REA
  StrSomme,ES,fonc,SommeString : Str20;
  For : RealToStringFormat;
  Win,Won,Win2,Win3,Win4 : WindowPtr;
```

```
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  TabImages : ARRAY [0..69] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3,ImgPtr4,ImgPtr5,ImgPtr6 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,ImgCount4,ImgCount5,ImgCount6,
                        Indice,i,I,X,Y,list6,list7 : CARDINAL;
  Res : BOOLEAN;
  SomPron,PPron,DerPron,UU : Str100;
  Phrase,Maxaf,U,UUUU  : Str20;
  Phrase2 : ARRAY [0..39] OF CHAR;
  In : CARDINAL;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree : REAL;
  DSR : DateStampRecord;

PROCEDURE GadgetHandlerEffBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Efface := TRUE ; DoneEff := TRUE;
           Trace := "Il confirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END ;
     1 : Efface := FALSE; DoneEff := TRUE;
           Trace := "Il infirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END ;
     2 :
  END;
END GadgetHandlerEffBS;


PROCEDURE GadgetHandlerFinBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN

  CASE Gad.GadgetID OF
     0 : Fini := TRUE ; DoneFin := TRUE;
           IF TableauChaine.Indice < 499
           THEN TableauChaine.Tab[TableauChaine.Indice] :=29;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick;
           TableauChaine.Indice := TableauChaine.Indice + 1;
           TableauChaine.Tab[TableauChaine.Indice] :=7;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
```

```
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il confirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END; |

    1 : Fini := FALSE; DoneFin := TRUE;
            IF TableauChaine.Indice < 499
            THEN TableauChaine.Tab[TableauChaine.Indice] :=29;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
                TableauChaine.Tab[TableauChaine.Indice] :=26;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END|
    2 :
  END;
END GadgetHandlerFinBS;

PROCEDURE GadgetHandlerRecBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN

  CASE Gad.GadgetID OF
    0 : Recom := TRUE ; DoneRec := TRUE;
            Trace := "Il confirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END |
    1 : Recom := FALSE; DoneRec := TRUE;
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END |
```

```
    2 :
   END;
END GadgetHandlerRecBS;

PROCEDURE GadgetHandlerBS (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
BEGIN
  IF OpenSpeech () THEN
  CASE Gad.GadgetID OF
     0 : ActivEff := TRUE;
         Nb200 := Nb200 + 1;
         SommePrecedente := Somme;
         Somme := Somme + 200.;
         IF Nb200 <= 5
         THEN
            Der.X := 170 + ((Nb200 - 1) * 20);
            Der.Y := 15;
            Der.Type := "BIL";
            Der.Last := 200.;
            AuMoinsUn := TRUE;
            Delay (25);
            DrawImage (Win^.RPort^,TabImages[50],Der.X,Der.Y)
         ELSE
            AuMoinsUn := FALSE;
            Der.Last := 200.
         END;
         IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("duh soh froh"));
           DerPron := "duh soh froh";
         END;
         wMove (Won^,7,1);
         wClrEndLine (Won^);
         ConvRealToString (Somme, StrSomme, 2, For);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,15,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);

         Trace := "Il appuie sur 200 Frs ";
         IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END ;
     1 : ActivEff := TRUE;
         Nb100 := Nb100 + 1;
         SommePrecedente := Somme;
         Somme := Somme + 100.;
         IF Nb100 <= 5
         THEN
            Der.X := 170 + ((Nb100 - 1) * 20);
            Der.Y := 31;
            Der.Type := "BIL";
            Der.Last := 100.;
            AuMoinsUn := TRUE;
            Delay (25);
            DrawImage (Win^.RPort^,TabImages[11],Der.X,Der.Y)
         ELSE
            AuMoinsUn := FALSE;
            Der.Last := 100.
         END;
         IF (TableauParametres[1] = 2) THEN
            Res := SayAndReturn (ADR("soh froh"));
            DerPron := "soh froh";
         END;
         wMove (Won^,7,1);
         wClrEndLine (Won^);
```

```
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 100 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
2 : ActivEff := TRUE;
    Nb50 := Nb50 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 50.;
    IF Nb50 <= 5
    THEN
       Der.X := 170 + ((Nb50 - 1) * 20);
       Der.Y := 47;
       Der.Type := "BIL";
       Der.Last := 50.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[12],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 50.
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("sekant froh"));
      DerPron := "sekant froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 50 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
3 : ActivEff := TRUE;
    Nb20 := Nb20 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 20.;
    IF Nb20 <= 5
    THEN
       Der.X := 170 + ((Nb20 - 1) * 20);
       Der.Y := 63;
       Der.Type := "BIL";
       Der.Last := 20.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[13],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 20.
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("vant froh"));
      DerPron := "vant froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
```

```
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 20 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END |
  4 : ActivEff := TRUE;
      Nb10 := Nb10 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 10.;
      IF Nb10 <= 7
      THEN
         Der.X := 170 + ((Nb10 - 1) * 18);
         Der.Y := 78;
         Der.Type := "PA";
         Der.Last := 10.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[19],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 10.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("dhe froh"));
        DerPron := "dhe froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 10 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END |
  5 : ActivEff := TRUE;
      Nb5 := Nb5 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 5.;
      IF Nb5 <= 7
      THEN
         Der.X := 170 + ((Nb5 - 1) * 18);
         Der.Y := 97;
         Der.Type := "PA";
         Der.Last := 5.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[10],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 5.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("senk froh"));
        DerPron := "senk froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
```

```
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 5 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  6 : ActivEff := TRUE;
      Nb2 := Nb2 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 2.;
      IF Nb2 <= 7
      THEN
         Der.X := 170 + ((Nb2 - 1) * 18);
         Der.Y := 113;
         Der.Type := "PA";
         Der.Last := 2.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[49],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 2.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("duh froh"));
        DerPron := "duh froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 2 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  7 : ActivEff := TRUE;
      Nb1 := Nb1 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 1.;
      IF Nb1 <= 7
      THEN
         Der.X := 170 + ((Nb1 - 1) * 18);
         Der.Y := 129;
         Der.Type := "PA";
         Der.Last := 1.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[17],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 1.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("an froh"));
        DerPron := "an froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
```

```
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 1 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  8 : ActivEff := TRUE;
      Nb05 := Nb05 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.5;
      IF Nb05 <= 7
      THEN
         Der.X := 170 + ((Nb05 - 1) * 18);
         Der.Y := 146;
         Der.Type := "PA";
         Der.Last := 0.5;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[7],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 0.5
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("sekant santeam"));
        DerPron := "sekant santeam";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 0.50 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  9 : ActivEff := TRUE;
      Nb02 := Nb02 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.2;
      IF Nb02 <= 7
      THEN
         Der.X := 170 + ((Nb02 - 1) * 18);
         Der.Y := 163;
         Der.Type := "PA";
         Der.Last := 0.2;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[18],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 0.2
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("vant santeam"));
        DerPron := "vant santeam";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
```

```
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 0.20 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
10: ActivEff := TRUE;
    Nb01 := Nb01 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 0.1;
    IF Nb01 <= 7
    THEN
       Der.X := 170 + ((Nb01 - 1) * 18);
       Der.Y := 180;
       Der.Type := "PA";
       Der.Last := 0.1;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[19],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 0.1
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("dhe santeam"));
      DerPron := "dhe santeam";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 0.1 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
11: ActivEff := TRUE;
    Nb005 := Nb005 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 0.05;
    IF Nb005 <= 7
    THEN
       Der.X := 170 + ((Nb005 - 1) * 18);
       Der.Y := 197;
       Der.Type := "PA";
       Der.Last := 0.05;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[10],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 0.05
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("senk santeam"));
      DerPron := "senk santeam";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
```

```
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      Trace := "Il appuie sur 0.05 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;

12:   IF ActivEff THEN
        IF TableauChaine.Indice < 499
        THEN TableauChaine.Tab[TableauChaine.Indice] :=27;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
            TableauChaine.Tab[TableauChaine.Indice] :=26;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
            TableauChaine.Indice := TableauChaine.Indice + 1;
        END;
        Trace := "Il demande d'effacer ";
        IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END;
        BeginGadgetList ();
        GadgetTypeReq := TRUE;
        GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                         RelVerify});
        AddGadgetTextButton (30,34, ADR("OUI"));
        AddGadgetTextButton (115,34, ADR("NON"));
        AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS EFFACER ?"));
        GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                   GadgetMutualExcludeSet {});
        GlEff := EndGadgetList ();
        InitRequester (ReqEff);
        WITH ReqEff DO
              OlderRequest := NIL;
              LeftEdge := 70;
              TopEdge := 8;
              Width := 180;
              Height := 50;
              ReqGadget := GlEff;
              ReqBorder := NIL;
```

```
        ReqText := NIL;
        Flags := RequesterFlagsSet {};
        BackFill := BYTE(10);
        ReqLayer := NIL;
        ImageBMap := NIL;
END;
IF Request (ReqEff, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("Voolay voo effahsay ?"));
   END;
   DoneEff := FALSE;
   WHILE NOT DoneEff DO
       Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
       LOOP
          MsgEff := GetMsg (Win^.UserPort^);
          IF (MsgEff = NIL) THEN EXIT; END;
          ProcIMsg (WpEff, MsgEff);
       END;
   END;
END;
FreeGadgetList (GlEff^);
Delay(25);
IF Efface THEN
  ActivEff := FALSE;
  IF ((CompareString (Der.Type,"BIL") = equal) AND (AuMoinsUn))
    THEN DrawImage (Win^.RPort^,TabImages[24],Der.X,Der.Y);
      IF (Der.X # 170) THEN
        IF Der.Last = 100.
          THEN DrawImage (Win^.RPort^,TabImages[11],Der.X - 20,Der.Y
          ELSIF Der.Last = 50.
          THEN DrawImage (Win^.RPort^,TabImages[12],Der.X - 20,Der.Y
          ELSIF Der.Last = 20.
          THEN DrawImage (Win^.RPort^,TabImages[13],Der.X - 20,Der.Y
          ELSIF Der.Last = 200.
          THEN DrawImage (Win^.RPort^,TabImages[50],Der.X - 20,Der.Y
        END
      END
    ELSIF ((CompareString (Der.Type,"P5F") = equal) AND (AuMoinsUn)
      THEN DrawImage (Win^.RPort^,TabImages[25],Der.X,Der.Y)
      ELSIF ((CompareString (Der.Type,"PA") = equal) AND (AuMoinsUn
        THEN DrawImage (Win^.RPort^,TabImages[26],Der.X,Der.Y)
  END;
  SommePrecedente := Somme;
  Somme := Somme - Der.Last;
  wMove (Won^,7,1);
  wClrEndLine (Won^);
  ConvRealToString (Somme, StrSomme, 2, For);
  PutStr (Won^,ADR(StrSomme));
  wMove (Won^,15,1);
  PutStr (Won^,ADR("Frs"));
  wSetCursor (Won^,FALSE);
  DerPron := "";
  IF Der.Last = 200.
   THEN  Nb200 := Nb200 - 1
   ELSIF  Der.Last = 100.
   THEN  Nb100 := Nb100 - 1
   ELSIF  Der.Last = 50.
   THEN  Nb50 := Nb50 - 1
   ELSIF  Der.Last = 20.
   THEN  Nb20 := Nb20 - 1
   ELSIF  Der.Last = 10.
   THEN  Nb10 := Nb10 - 1
   ELSIF  Der.Last = 5.
   THEN  Nb5 := Nb5 - 1
   ELSIF  Der.Last = 2.
   THEN  Nb2 := Nb2 - 1
```

```
        ELSIF  Der.Last = 1.
        THEN  Nb1 := Nb1 - 1
        ELSIF  Der.Last = 0.5
        THEN  Nb05 := Nb05 - 1
        ELSIF  Der.Last = 0.2
        THEN  Nb02 := Nb02 - 1
        ELSIF  Der.Last = 0.1
        THEN  Nb01 := Nb01 - 1
        ELSIF Der.Last = 0.05
        THEN  Nb005 := Nb005 - 1
        ELSE
       END;
     END
   END ;
13 :Trace := "Il appuie sur OK ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END;
   BeginGadgetList ();
   GadgetTypeReq := TRUE;
   GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                    RelVerify});
   AddGadgetTextButton (30,34, ADR("OUI"));
   AddGadgetTextButton (115,34, ADR("NON"));
   AddGadgetTextButton (20,9, ADR("AVEZ-VOUS FINI ?"));
   GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
              GadgetMutualExcludeSet {});
   GlFin := EndGadgetList ();
   InitRequester (ReqFin);
   WITH ReqFin DO
         OlderRequest := NIL;
         LeftEdge := 70;
         TopEdge := 8;
         Width := 180;
         Height := 50;
         ReqGadget := GlFin;
         ReqBorder := NIL;
         ReqText := NIL;
         Flags := RequesterFlagsSet {};
         BackFill := BYTE(10);
         ReqLayer := NIL;
         ImageBMap := NIL;
   END;
   IF Request (ReqFin, Win^) THEN
      IF (TableauParametres[1] = 2) THEN
          Res := SayAndReturn (ADR("ahvay voo feeny ?"));
      END;
      DoneFin := FALSE;
      WHILE NOT DoneFin DO
          Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
          LOOP
             MsgFin := GetMsg (Win^.UserPort^);
             IF (MsgFin = NIL) THEN EXIT; END;
             ProcIMsg (WpFin, MsgFin);
          END;
      END;
   END;
   FreeGadgetList (GlFin^);
   Delay(25);
   IF Fini THEN Done := TRUE END;
14 :  IF TableauChaine.Indice < 499
     THEN TableauChaine.Tab[TableauChaine.Indice] :=28;
               DateStamp(DSR);
               ActMinute := SHORT(DSR.dsMinute);
               ActTick := SHORT(DSR.dsTick);
```

```
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
        TableauChaine.Indice := TableauChaine.Indice + 1;
        TableauChaine.Tab[TableauChaine.Indice] :=26;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
        TableauChaine.Indice := TableauChaine.Indice + 1;
END;
Trace := "Il appuie sur RECOMMENCER ";
IF TableauTraces.Indice <= 499 THEN
        CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
        TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                  RelVerify});
AddGadgetTextButton (30,34, ADR("OUI"));
AddGadgetTextButton (115,34, ADR("NON"));
AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS RECOMMENCER ?"));
GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
            GadgetMutualExcludeSet {});
G1Rec := EndGadgetList ();
InitRequester (ReqRec);
WITH ReqRec DO
        OlderRequest := NIL;
        LeftEdge := 50;
        TopEdge := 8;
        Width := 220;
        Height := 50;
        ReqGadget := G1Rec;
        ReqBorder := NIL;
        ReqText := NIL;
        Flags := RequesterFlagsSet {};
        BackFill := BYTE(10);
        ReqLayer := NIL;
        ImageBMap := NIL;
END;
IF Request (ReqRec, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("Vooley voo ra commohsay ?"));
   END;
   DoneRec := FALSE;
   WHILE NOT DoneRec DO
     Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
     LOOP
        MsgRec := GetMsg (Win^.UserPort^);
        IF (MsgRec = NIL) THEN EXIT; END;
        ProcIMsg (WpRec, MsgRec);
```

```
      END;
  END;
END;
FreeGadgetList (GlRec^);
Delay(25);
IF Recom THEN
      DrawImage (Win^.RPort^,TabImages[24],170,15);
      DrawImage (Win^.RPort^,TabImages[24],190,15);
      DrawImage (Win^.RPort^,TabImages[24],210,15);
      DrawImage (Win^.RPort^,TabImages[24],230,15);
      DrawImage (Win^.RPort^,TabImages[24],250,15);
      DrawImage (Win^.RPort^,TabImages[24],170,31);
      DrawImage (Win^.RPort^,TabImages[24],190,31);
      DrawImage (Win^.RPort^,TabImages[24],210,31);
      DrawImage (Win^.RPort^,TabImages[24],230,31);
      DrawImage (Win^.RPort^,TabImages[24],250,31);
      DrawImage (Win^.RPort^,TabImages[24],170,47);
      DrawImage (Win^.RPort^,TabImages[24],190,47);
      DrawImage (Win^.RPort^,TabImages[24],210,47);
      DrawImage (Win^.RPort^,TabImages[24],230,47);
      DrawImage (Win^.RPort^,TabImages[24],250,47);
      DrawImage (Win^.RPort^,TabImages[24],170,63);
      DrawImage (Win^.RPort^,TabImages[24],190,63);
      DrawImage (Win^.RPort^,TabImages[24],210,63);
      DrawImage (Win^.RPort^,TabImages[24],230,63);
      DrawImage (Win^.RPort^,TabImages[24],250,63);
      DrawImage (Win^.RPort^,TabImages[26],170,78);
      DrawImage (Win^.RPort^,TabImages[26],188,78);
      DrawImage (Win^.RPort^,TabImages[26],206,78);
      DrawImage (Win^.RPort^,TabImages[26],224,78);
      DrawImage (Win^.RPort^,TabImages[26],242,78);
      DrawImage (Win^.RPort^,TabImages[26],260,78);
      DrawImage (Win^.RPort^,TabImages[26],278,78);
      DrawImage (Win^.RPort^,TabImages[26],170,97);
      DrawImage (Win^.RPort^,TabImages[26],188,97);
      DrawImage (Win^.RPort^,TabImages[26],206,97);
      DrawImage (Win^.RPort^,TabImages[26],224,97);
      DrawImage (Win^.RPort^,TabImages[26],242,97);
      DrawImage (Win^.RPort^,TabImages[26],260,97);
      DrawImage (Win^.RPort^,TabImages[26],278,97);
      DrawImage (Win^.RPort^,TabImages[26],170,113);
      DrawImage (Win^.RPort^,TabImages[26],188,113);
      DrawImage (Win^.RPort^,TabImages[26],206,113);
      DrawImage (Win^.RPort^,TabImages[26],224,113);
      DrawImage (Win^.RPort^,TabImages[26],242,113);
      DrawImage (Win^.RPort^,TabImages[26],260,113);
      DrawImage (Win^.RPort^,TabImages[26],278,113);
      DrawImage (Win^.RPort^,TabImages[26],170,129);
      DrawImage (Win^.RPort^,TabImages[26],188,129);
      DrawImage (Win^.RPort^,TabImages[26],206,129);
      DrawImage (Win^.RPort^,TabImages[26],224,129);
      DrawImage (Win^.RPort^,TabImages[26],242,129);
      DrawImage (Win^.RPort^,TabImages[26],260,129);
      DrawImage (Win^.RPort^,TabImages[26],278,129);
      DrawImage (Win^.RPort^,TabImages[26],170,146);
      DrawImage (Win^.RPort^,TabImages[26],188,146);
      DrawImage (Win^.RPort^,TabImages[26],206,146);
      DrawImage (Win^.RPort^,TabImages[26],224,146);
      DrawImage (Win^.RPort^,TabImages[26],242,146);
      DrawImage (Win^.RPort^,TabImages[26],260,146);
      DrawImage (Win^.RPort^,TabImages[26],278,146);
      DrawImage (Win^.RPort^,TabImages[26],170,163);
      DrawImage (Win^.RPort^,TabImages[26],188,163);
      DrawImage (Win^.RPort^,TabImages[26],206,163);
      DrawImage (Win^.RPort^,TabImages[26],224,163);
      DrawImage (Win^.RPort^,TabImages[26],242,163);
```

```
                DrawImage (Win^.RPort^,TabImages[26],260,163);
                DrawImage (Win^.RPort^,TabImages[26],278,163);
                DrawImage (Win^.RPort^,TabImages[26],170,180);
                DrawImage (Win^.RPort^,TabImages[26],188,180);
                DrawImage (Win^.RPort^,TabImages[26],206,180);
                DrawImage (Win^.RPort^,TabImages[26],224,180);
                DrawImage (Win^.RPort^,TabImages[26],242,180);
                DrawImage (Win^.RPort^,TabImages[26],260,180);
                DrawImage (Win^.RPort^,TabImages[26],278,180);
                DrawImage (Win^.RPort^,TabImages[26],170,197);
                DrawImage (Win^.RPort^,TabImages[26],188,197);
                DrawImage (Win^.RPort^,TabImages[26],206,197);
                DrawImage (Win^.RPort^,TabImages[26],224,197);
                DrawImage (Win^.RPort^,TabImages[26],242,197);
                DrawImage (Win^.RPort^,TabImages[26],260,197);
                DrawImage (Win^.RPort^,TabImages[26],278,197);
                Somme := 0.;
                wMove (Won^,7,1);
                wClrEndLine (Won^);
                ConvRealToString (Somme, StrSomme, 2, For);
                PutStr (Won^,ADR(StrSomme));
                wMove (Won^,15,1);
                PutStr (Won^,ADR("Frs"));
                wSetCursor (Won^,FALSE);
                ES := "";
                SommeMax := Max;
                Nb200 := 0;
                Nb100 := 0;
                Nb50 := 0;
                Nb20 := 0;
                Nb10 := 0;
                Nb5 := 0;
                Nb2 := 0;
                Nb1 := 0;
                Nb05 := 0;
                Nb02 := 0;
                Nb01 := 0;
                Nb005 := 0;
                Der.Last := 0.;
                Der.X := 150;
                AuMoinsUn := FALSE;
                ActivEff := FALSE;
                For := Decimal;
                CopyString (DerPron,PPron);

            END;

        15:   IF (TableauParametres[1] = 2) THEN
                Res := SayAndReturn (ADR(DerPron));
              END;
    END;
CloseSpeech ();
END;
END GadgetHandlerBS;


PROCEDURE CorrPMBI (VAR Phrase : ARRAY OF CHAR;
                    VAR PhrasePron : ARRAY OF CHAR );



BEGIN

    IF TableauChaine.Indice <= 499
    THEN TableauChaine.Tab[TableauChaine.Indice] :=26;
                    DateStamp(DSR);
```

```
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
        TableauChaine.Indice := TableauChaine.Indice + 1;
    END;

    CopyString (PPron,PhrasePron);
    QR := FALSE;
    Somme := 0.;
    SommeMax := Max;
    Nb200 := 0;
    Nb100 := 0;
    Nb50 := 0;
    Nb20 := 0;
    Nb10 := 0;
    Nb5 := 0;
    Nb2 := 0;
    Nb1 := 0;
    Nb05 := 0;
    Nb02 := 0;
    Nb01 := 0;
    Nb005 := 0;
    Der.Last := 0.;
    Der.X := 150;
    AuMoinsUn := FALSE;
    ActivEff := FALSE;
    For := Decimal;

(*    IF OpenSpeech () THEN *)
    WITH Wp DO
      procGadgetUp := GadgetHandlerBS;
    END;
    WITH WpEff DO
      procGadgetUp := GadgetHandlerEffBS;
    END;
    WITH WpFin DO
      procGadgetUp := GadgetHandlerFinBS;
    END;
    WITH WpRec DO
      procGadgetUp := GadgetHandlerRecBS;
    END;
    Scr := CreateScreen (320,230,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 0FCAH;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
```

```
    cmap[15] := 0C08H;
    cmap[16] := 0620H;
    cmap[17] := 0E52H;
    cmap[18] := 0A52H;
    cmap[19] := 0FCAH;
    cmap[20] := 0333H;
    cmap[21] := 0444H;
    cmap[22] := 0555H;
    cmap[23] := 0666H;
    cmap[24] := 0777H;
    cmap[25] := 0888H;
    cmap[26] := 0999H;
    cmap[27] := 0AAAH;
    cmap[28] := 0CCCH;
    cmap[29] := 0DDDH;
    cmap[30] := 0EEEH;
    cmap[31] := 0FFFH;
    LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

    ImgPtr := FindImageTable(33333333H, ImgCount);
    IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
      Indice := 0;
      WHILE (Indice <= ImgCount-1) DO
        WITH TabImages[Indice] DO
          LeftEdge    := 0;
          TopEdge     := 0;
          Width       := ImgPtr^[Indice].Width;
          Height      := ImgPtr^[Indice].Height;
          Depth       := ImgPtr^[Indice].Depth;
          ImageData   := ImgPtr^[Indice].Data;
          PlanePick   := BYTE(31);
          PlaneOnOff  := BYTE(31);
          NextImage := NIL;
        END;
        Indice := Indice + 1;
      END;
    END;
    ImgPtr2 := FindImageTable(44444444H, ImgCount2);
     IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
      WHILE (Indice-ImgCount <= ImgCount2-1) DO
        WITH TabImages[Indice] DO
          LeftEdge    := 0;
          TopEdge     := 0;
          Width       := ImgPtr2^[Indice-ImgCount].Width;
          Height      := ImgPtr2^[Indice-ImgCount].Height;
          Depth       := ImgPtr2^[Indice-ImgCount].Depth;
          ImageData   := ImgPtr2^[Indice-ImgCount].Data;
          PlanePick   := BYTE(31);
          PlaneOnOff  := BYTE(31);
          NextImage := NIL;
        END;
        Indice := Indice + 1;
      END;
     END;
     ImgPtr3 := FindImageTable(77777777H, ImgCount3);
     IF (ImgPtr3 # NIL) AND (ImgCount3 # 0) THEN
        WITH TabImages[Indice] DO
          LeftEdge    := 0;
          TopEdge     := 0;
          Width       := ImgPtr3^[9].Width;
          Height      := ImgPtr3^[9].Height;
          Depth       := ImgPtr3^[9].Depth;
          ImageData   := ImgPtr3^[9].Data;
          PlanePick   := BYTE(31);
          PlaneOnOff  := BYTE(31);
          NextImage := NIL;
```

```
    END;
  END;
  Indice := Indice + 1;

  ImgPtr4 := FindImageTable(66666666H, ImgCount4);
  IF (ImgPtr4 # NIL) AND (ImgCount4 # 0) THEN
    list6 := 3;
    WHILE (list6 <= ImgCount4-1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr4^[list6].Width;
        Height      := ImgPtr4^[list6].Height;
        Depth       := ImgPtr4^[list6].Depth;
        ImageData   := ImgPtr4^[list6].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      list6 := list6 + 1;
      Indice := Indice + 1;
    END;
END;
ImgPtr5 := FindImageTable(11111114H, ImgCount5);
 IF (ImgPtr5 # NIL) AND (ImgCount5 # 0) THEN
  list7 := 3;
  WHILE (list7 <= ImgCount5-1) DO
     WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr5^[list7].Width;
        Height      := ImgPtr5^[list7].Height;
        Depth       := ImgPtr5^[list7].Depth;
        ImageData   := ImgPtr5^[list7].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      list7 := list7 + 1;
      Indice := Indice + 1;
    END;
END;
ImgPtr6 := FindImageTable(22222223H, ImgCount6);
 IF (ImgPtr6 # NIL) AND (ImgCount6 # 0) THEN
  list7 := 0;
  Indice := 46;
  WHILE (list7 <= ImgCount6-1) DO
     WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr6^[list7].Width;
        Height      := ImgPtr6^[list7].Height;
        Depth       := ImgPtr6^[list7].Depth;
        ImageData   := ImgPtr6^[list7].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      list7 := list7 + 1;
      Indice := Indice + 1;
    END;
END;
    BeginGadgetList();
    AddGadgetImageButton (10,10,TabImages[48]);
    AddGadgetImageButton (10,47,TabImages[0]);
    AddGadgetImageButton (10,84,TabImages[1]);
```

```
AddGadgetImageButton (10,121,TabImages[2]);
AddGadgetImageButton (88,5,TabImages[47]);
AddGadgetImageButton (85,34,TabImages[4]);
AddGadgetImageButton (87,70,TabImages[46]);
AddGadgetImageButton (90,101,TabImages[6]);
AddGadgetImageButton (92,125,TabImages[7]);
AddGadgetImageButton (91,143,TabImages[8]);
AddGadgetImageButton (92,164,TabImages[9]);
AddGadgetImageButton (93,183,TabImages[10]);
AddGadgetImageButton (4,200,TabImages[20]);
AddGadgetImageButton (121,200,TabImages[21]);
AddGadgetImageButton (63,200,TabImages[36]);
IF TableauParametres[1] = 2 THEN
  AddGadgetImageButton (6,160,TabImages[35]);
END;
Gl := EndGadgetList();
IF (Gl # NIL) THEN
   Win := CreateWindow (0,0,320,230,WIDCMP,WFlags,Gl,Scr,NIL);
   Win2 := CreateWindow (134,72,23,102,WIDCMP,WFlags2,NIL,Scr,NIL);
   Won := CreateWindow (168,212,152,15,WIDCMP,WFlags,NIL,Scr,NIL);
   IF (Win # NIL) AND (Won # NIL)  THEN

    IF CreateConsole (Won^) THEN
      wClrScr (Won^);
      PutStr (Won^,ADR("Total"));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      DrawImage (Win^.RPort^,TabImages[28],165,0);
      DrawImage (Win^.RPort^,TabImages[28],165,100);
      IF NOT QR THEN
        DrawImage (Win^.RPort^,TabImages[43],0,0);
        DrawImage (Win^.RPort^,TabImages[43],130,0);
        DrawImage (Win^.RPort^,TabImages[43],190,0);
        DrawImage (Win^.RPort^,TabImages[43],0,227);
        DrawImage (Win^.RPort^,TabImages[43],130,227);
        DrawImage (Win^.RPort^,TabImages[43],190,227);
        DrawImage (Win^.RPort^,TabImages[45],0,0);
        DrawImage (Win^.RPort^,TabImages[45],0,100);
        DrawImage (Win^.RPort^,TabImages[45],317,0);
        DrawImage (Win^.RPort^,TabImages[45],317,100);
        DrawImage (Win^.RPort^,TabImages[42],123,3);
        DrawImage (Win^.RPort^,TabImages[45],165,0);
        DrawImage (Win^.RPort^,TabImages[45],165,100);
        DrawImage (Win^.RPort^,TabImages[39],175,5);

      END;
      IF (CompareString (PhrasePron,"Cobya ah vay voo ra su darjo ?")
            = equal)
      THEN
        DrawImage (Win^.RPort^,TabImages[31],0,0);
        DrawImage (Win^.RPort^,TabImages[31],130,0);
        DrawImage (Win^.RPort^,TabImages[31],190,0);
        DrawImage (Win^.RPort^,TabImages[31],0,227);
        DrawImage (Win^.RPort^,TabImages[31],130,227);
        DrawImage (Win^.RPort^,TabImages[31],190,227);
        DrawImage (Win^.RPort^,TabImages[29],0,0);
        DrawImage (Win^.RPort^,TabImages[29],0,100);
        DrawImage (Win^.RPort^,TabImages[29],317,0);
        DrawImage (Win^.RPort^,TabImages[29],317,100);
        DrawImage (Win^.RPort^,TabImages[33],123,3);
        DrawImage (Win^.RPort^,TabImages[29],165,0);
        DrawImage (Win^.RPort^,TabImages[29],165,100);
        DrawImage (Win^.RPort^,TabImages[41],175,5);

      ELSIF (CompareString (PhrasePron,"Cobya ah vay voo daypensay ?
```

```
                = equal)
        THEN
          DrawImage (Win^.RPort^,TabImages[32],0,0);
          DrawImage (Win^.RPort^,TabImages[32],130,0);
          DrawImage (Win^.RPort^,TabImages[32],190,0);
          DrawImage (Win^.RPort^,TabImages[32],0,227);
          DrawImage (Win^.RPort^,TabImages[32],130,227);
          DrawImage (Win^.RPort^,TabImages[32],190,227);
          DrawImage (Win^.RPort^,TabImages[30],0,0);
          DrawImage (Win^.RPort^,TabImages[30],0,100);
          DrawImage (Win^.RPort^,TabImages[30],317,0);
          DrawImage (Win^.RPort^,TabImages[30],317,100);
          DrawImage (Win^.RPort^,TabImages[34],123,3);
          DrawImage (Win^.RPort^,TabImages[30],165,0);
          DrawImage (Win^.RPort^,TabImages[30],165,100);
          DrawImage (Win^.RPort^,TabImages[40],175,5);

        END;
        IF TableauParametres[1] = 2 THEN
          IF OpenSpeech () THEN
                Res := SayAndReturn (ADR(PhrasePron));
                CopyString (DerPron,PhrasePron);
                CloseSpeech ()
          END;
        END;

        SA := Sactuel;
        In :=1;
        WHILE (In <= TabNombreInit [0]) DO

Nb200 := Nb200 + 1;
SommePrecedente := Somme;
Somme := Somme + 200.;
IF Nb200 <= 5
THEN
   Der.X := 170 + ((Nb200 - 1) * 20);
   Der.Y := 15;
   Der.Type := "BIL";
   Der.Last := 200.;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[50],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 200.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

      In := In+1;
     END;

        In := 1;
        WHILE (In <= TabNombreInit [1]) DO

Nb100 := Nb100 + 1;
SommePrecedente := Somme;
Somme := Somme + 100.;
IF Nb100 <= 5
THEN
   Der.X := 170 + ((Nb100 - 1) * 20);
```

```
   Der.Y := 31;
   Der.Type := "BIL";
   Der.Last := 100.;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[11],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 100.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
    END;

          In := 1;
          WHILE (In <=TabNombreInit [2]) DO

Nb50 := Nb50 + 1;
SommePrecedente := Somme;
Somme := Somme + 50.;
IF Nb50 <= 5
THEN
   Der.X := 170 + ((Nb50 - 1) * 20);
   Der.Y := 47;
   Der.Type := "BIL";
   Der.Last := 50.;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[12],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 50.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

    In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [3]) DO

Nb20 := Nb20 + 1;
SommePrecedente := Somme;
Somme := Somme + 20.;
IF Nb20 <= 5
THEN
   Der.X := 170 + ((Nb20 - 1) * 20);
   Der.Y := 63;
   Der.Type := "BIL";
   Der.Last := 20.;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[13],Der.X,Der.Y)
```

```
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 20.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

  In := In+1;
    END;

      In :=1;
      WHILE (In <=TabNombreInit [4]) DO


Nb10 := Nb10 + 1;
SommePrecedente := Somme;
Somme := Somme + 10.;
IF Nb10 <= 7
THEN
   Der.X := 170 + ((Nb10 - 1) * 18);
   Der.Y := 78;
   Der.Type := "PA";
   Der.Last := 10.;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[19],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 10.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);
  In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [5]) DO


Nb5 := Nb5 + 1;
SommePrecedente := Somme;
Somme := Somme + 5.;
IF Nb5 <= 7
THEN
   Der.X := 170 + ((Nb5 - 1) * 18);
   Der.Y := 97;
   Der.Type := "PA";
   Der.Last := 5.;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[10],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 5.
END;
wMove (Won^,7,1);
```

```
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

  In := In+1;
END;

     In :=1;
     WHILE (In <= TabNombreInit [6]) DO


Nb2 := Nb2 + 1;
SommePrecedente := Somme;
Somme := Somme + 2.;
IF Nb2 <= 7
THEN
   Der.X := 170 + ((Nb2 - 1) * 18);
   Der.Y := 113;
   Der.Type := "PA";
   Der.Last := 2.;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[49],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 2.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

   In := In+1;
END;

    In := 1;
    WHILE (In <= TabNombreInit [7]) DO


Nb1 := Nb1 + 1;
SommePrecedente := Somme;
Somme := Somme + 1.;
IF Nb1 <= 7
THEN
   Der.X := 170 + ((Nb1 - 1) * 18);
   Der.Y := 129;
   Der.Type := "PA";
   Der.Last := 1.;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[17],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 1.
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
```

```
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

 In := In+1;
END;

   In := 1;
   WHILE (In <= TabNombreInit [8]) DO


NbO5 := NbO5 + 1;
SommePrecedente := Somme;
Somme := Somme + 0.5;
IF NbO5 <= 7
THEN
   Der.X := 170 + ((NbO5 - 1) * 18);
   Der.Y := 146;
   Der.Type := "PA";
   Der.Last := 0.5;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[7],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 0.5
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

  In :=In+1;
END;

   In :=1;
   WHILE (In <= TabNombreInit [9]) DO


NbO2 := NbO2 + 1;
SommePrecedente := Somme;
Somme := Somme + 0.2;
IF NbO2 <= 7
THEN
   Der.X := 170 + ((NbO2 - 1) * 18);
   Der.Y := 163;
   Der.Type := "PA";
   Der.Last := 0.2;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[18],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 0.2
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

  In := In+1;
```

```
END;

    In :=1;
    WHILE (In <= TabNombreInit [10]) DO

Nb01 := Nb01 + 1;
SommePrecedente := Somme;
Somme := Somme + 0.1;
IF Nb01 <= 7
THEN
   Der.X := 170 + ((Nb01 - 1) * 18);
   Der.Y := 180;
   Der.Type := "PA";
   Der.Last := 0.1;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[19],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 0.1
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

 In := In +1;
END;

    In := 1;
    WHILE (In <= TabNombreInit [11] ) DO


Nb005 := Nb005 + 1;
SommePrecedente := Somme;
Somme := Somme + 0.05;
IF Nb005 <= 7
THEN
   Der.X := 170 + ((Nb005 - 1) * 18);
   Der.Y := 197;
   Der.Type := "PA";
   Der.Last := 0.05;
   AuMoinsUn := TRUE;
   Delay (25);
   DrawImage (Win^.RPort^,TabImages[10],Der.X,Der.Y)
ELSE
   AuMoinsUn := FALSE;
   Der.Last := 0.05
END;
wMove (Won^,7,1);
wClrEndLine (Won^);
ConvRealToString (Somme, StrSomme, 2, For);
PutStr (Won^,ADR(StrSomme));
wMove (Won^,15,1);
PutStr (Won^,ADR("Frs"));
wSetCursor (Won^,FALSE);

In := In +1;
END;

      Done := FALSE;
      WHILE (NOT Done) DO
        Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
```

```
                LOOP
                  Msg := GetMsg(Win^.UserPort^);
                  IF (Msg = NIL) THEN EXIT; END;
                  ProcIMsg (Wp, Msg);
                END;
              END;
              DeleteConsole (Won^);
              ELSE WriteString ("Console non créée")
            END;
          Sinit := Somme;

          CloseWindow(Won^);
          CloseWindow(Win^);
          END;
          FreeGadgetList (Gl^);
        END;
      CloseScreen(Scr^);
    END;

  TabNombreInit[0] := Nb200;
  TabNombreInit[1] := Nb100;
  TabNombreInit[2] := Nb50;
  TabNombreInit[3] := Nb20;
  TabNombreInit[4] := Nb10;
  TabNombreInit[5] := Nb5;
  TabNombreInit[6] := Nb2;
  TabNombreInit[7] := Nb1;
  TabNombreInit[8] := Nb05;
  TabNombreInit[9] := Nb02;
  TabNombreInit[10] := Nb01;
  TabNombreInit[11] := Nb005;

END CorrPMBI;

(*  VAR p,pp : ARRAY [0..40] OF CHAR;

BEGIN
    p := "Combien avez-vous d'argent ?";
    pp := "Cobya ah vay voo darjo ?";
    CorrPMBI (p,pp); *)

END CorrPMBillets.
```

```
IMPLEMENTATION MODULE SSBillets;

FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM AmigaDOS IMPORT DateStamp,DateStampRecord;
FROM MathLib0 IMPORT entier,real;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr,BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetImageButton, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
       StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100,TabNombreInit,
                TempsSIBillets,TableauTraces,TRecettes,TDepenses,
                               TableauParametres,Sinit,Max,TabNombre,Sactuel,
                               TableauChaine,AncMinute,AncTick,TableauTemps;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                               ConvStringToReal;
FROM Utilitaires IMPORT ConvPrononcable;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;
VAR
  Done,DoneOK,DoneRec,Recom : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff,QR : BOOLEAN;
  G1,G1Eff,G1Qui,G1Fin,G1OK,G1Rec : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK,WpRec : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK,ReqRec : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK,MsgRec : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb200,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
  Trace : Str40;
  Der : Dernier;
  Somme,SommePrecedente, SommeMax,ValeurMax,ValeurPrec,SomAct,SA,SAP,UUU : RE
  StrSomme,ES,fonc,SommeString : Str20;
  For : RealToStringFormat;
  Win,Won,Win2,Win3,Win4 : WindowPtr;
```

```
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  TabImages : ARRAY [0..69] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3,ImgPtr4,ImgPtr5,ImgPtr6 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,ImgCount4,ImgCount5,ImgCount6,
                    Indice,i,I,X,Y,list6,list7 : CARDINAL;
  Res : BOOLEAN;
  SomPron,PPron,DerPron,UU : Str100;
  Phrase,Maxaf,U,UUUU  : Str20;
  Phrase2 : ARRAY [0..39] OF CHAR;
  ActMinute,ActTick,ent,ActTick2 : INTEGER;
  ree,ree2 : REAL;
  DSR : DateStampRecord;
  IndTemps,II : CARDINAL;
  Nbillets : INTEGER;
  NminBEG,NticksBEG,NticksEND,NminEND,TempsInterm : ARRAY [1..20] OF INTEGER;

PROCEDURE GadgetHandlerEffBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Efface := TRUE ; DoneEff := TRUE;
           IF TableauChaine.Indice <= 499
           THEN IF NOT QR
                THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                           = equal)
                     THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                          TableauChaine.Indice := TableauChaine.Indice + 1;
                     ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
```

```
                    TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
            END;
        END;
        Trace := "Il confirme la demande ";
        IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END |
  1 : Efface := FALSE; DoneEff := TRUE;
        IF TableauChaine.Indice <= 499
        THEN IF NOT QR
            THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
            ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                        = equal)
                 THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                 ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                DateStamp(DSR);
                ActMinute := SHORT(DSR.dsMinute);
                ActTick := SHORT(DSR.dsTick);
                ActTick2 := ActTick;
                IF ActMinute > AncMinute
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                      TableauChaine.Indice := TableauChaine.Indice + 1;
                 END;
            END;
        END;
        Trace := "Il infirme la demande ";
        IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END |
  2 :
END;
```

```
END GadgetHandlerEffBS;

PROCEDURE GadgetHandlerQuiBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : Quitter := TRUE ; DoneQui := TRUE;
           IF TableauChaine.Indice <= 499
           THEN TableauChaine.Tab[TableauChaine.Indice] := 7;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
           Trace := "Il confirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END ;
     1 : Quitter := FALSE; DoneQui := TRUE;
           IF TableauChaine.Indice <= 499
           THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                          = equal)
                THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                     TableauChaine.Indice := TableauChaine.Indice + 1;
                END;
            END;
           Trace := "Il infirme la demande ";
           IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
           END ;
```

```
    2 :
  END;
END GadgetHandlerQuiBS;

PROCEDURE GadgetHandlerFinBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN

  CASE Gad.GadgetID OF
     0 : Fini := TRUE ; DoneFin := TRUE;
         IF TableauChaine.Indice <= 499
         THEN TableauChaine.Tab[TableauChaine.Indice] := 7;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
              TableauChaine.Indice := TableauChaine.Indice + 1;
         END;
         Trace := "Il confirme la demande ";
         IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END; |

     1 : Fini := FALSE; DoneFin := TRUE;
         IF TableauChaine.Indice <= 499
         THEN IF NOT QR
              THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
              ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                           = equal)
                   THEN TableauChaine.Tab [TableauChaine.Indice] := 20;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                        TableauChaine.Indice := TableauChaine.Indice + 1;
                   ELSE TableauChaine.Tab [TableauChaine.Indice] := 12;
                  DateStamp(DSR);
```

```
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
                          TableauChaine.Indice := TableauChaine.Indice + 1;
                     END;
               END;
           END;
          Trace := "Il infirme la demande ";
          IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
          END;
     2 :
  END;
END GadgetHandlerFinBS;
PROCEDURE GadgetHandlerRecBS (VAR w:Window; VAR Msg :MsgData; VAR Gad:Gadget);
BEGIN

  CASE Gad.GadgetID OF
     0 : Recom := TRUE ; DoneRec := TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
          Trace := "Il confirme la demande ";
          IF TableauTraces.Indice <= 499 THEN
               CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
               TableauTraces.Indice :=  TableauTraces.Indice + 1;
          END ;
     1 : Recom := FALSE; DoneRec := TRUE;
          IF TableauChaine.Indice <= 499
          THEN TableauChaine.Tab[TableauChaine.Indice] := 3;
                    DateStamp(DSR);
                    ActMinute := SHORT(DSR.dsMinute);
                    ActTick := SHORT(DSR.dsTick);
                    ActTick2 := ActTick;
                    IF ActMinute > AncMinute
                    THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                    END;
                    ent := ActTick - AncTick;
                    ree := real(ent);
                    TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                    AncMinute := ActMinute;
                    AncTick := ActTick2;
               TableauChaine.Indice := TableauChaine.Indice + 1;
          END;
```

```
            Trace := "Il infirme la demande ";
            IF TableauTraces.Indice <= 499 THEN
                 CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                 TableauTraces.Indice :=  TableauTraces.Indice + 1;
            END ;
      2 :
    END;
END GadgetHandlerRecBS;

PROCEDURE GadgetHandlerBS (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
BEGIN
  IF OpenSpeech () THEN
  CASE Gad.GadgetID OF
     0 : DateStamp(DSR);
         NminEND[IndTemps] := SHORT(DSR.dsMinute);
         NticksEND[IndTemps] := SHORT(DSR.dsTick);
         Nbillets := Nbillets + 1;
         ActivEff := TRUE;
         Nb200 := Nb200 + 1;
         SommePrecedente := Somme;
         Somme := Somme + 200.;
         IF Nb200 <= 5
         THEN
            Der.X := 170 + ((Nb200 - 1) * 20);
            Der.Y := 15;
            Der.Type := "BIL";
            Der.Last := 200.;
            AuMoinsUn := TRUE;
            Delay (25);
            DrawImage (Win^.RPort^,TabImages[50],Der.X,Der.Y)
         ELSE
            AuMoinsUn := FALSE;
            Der.Last := 200.
         END;
         IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("duh soh froh"));
           DerPron := "duh soh froh";
         END;
         wMove (Won^,7,1);
         wClrEndLine (Won^);
         ConvRealToString (Somme, StrSomme, 2, For);
         PutStr (Won^,ADR(StrSomme));
         wMove (Won^,15,1);
         PutStr (Won^,ADR("Frs"));
         wSetCursor (Won^,FALSE);

         Trace := "Il appuie sur 200 Frs ";
         IF TableauTraces.Indice <= 499 THEN
                CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
                TableauTraces.Indice :=  TableauTraces.Indice + 1;
         END ;
     1 : DateStamp(DSR);
         NminEND[IndTemps] := SHORT(DSR.dsMinute);
         NticksEND[IndTemps] := SHORT(DSR.dsTick);
         Nbillets := Nbillets + 1;
         ActivEff := TRUE;
         Nb100 := Nb100 + 1;
         SommePrecedente := Somme;
         Somme := Somme + 100.;
         IF Nb100 <= 5
         THEN
            Der.X := 170 + ((Nb100 - 1) * 20);
            Der.Y := 31;
            Der.Type := "BIL";
            Der.Last := 100.;
```

```
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,TabImages[11],Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 100.
   END;
   IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("soh froh"));
      DerPron := "soh froh";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 2, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);

   Trace := "Il appuie sur 100 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END ;
2 : DateStamp(DSR);
   NminEND[IndTemps] := SHORT(DSR.dsMinute);
   NticksEND[IndTemps] := SHORT(DSR.dsTick);
   Nbillets := Nbillets + 1;
   ActivEff := TRUE;
   Nb50 := Nb50 + 1;
   SommePrecedente := Somme;
   Somme := Somme + 50.;
   IF Nb50 <= 5
   THEN
      Der.X := 170 + ((Nb50 - 1) * 20);
      Der.Y := 47;
      Der.Type := "BIL";
      Der.Last := 50.;
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,TabImages[12],Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 50.
   END;
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("sekant froh"));
     DerPron := "sekant froh";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 2, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);

   Trace := "Il appuie sur 50 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END ;
3 : DateStamp(DSR);
   NminEND[IndTemps] := SHORT(DSR.dsMinute);
   NticksEND[IndTemps] := SHORT(DSR.dsTick);
   Nbillets := Nbillets + 1;
```

```
      ActivEff := TRUE;
      Nb20 := Nb20 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 20.;
      IF Nb20 <= 5
      THEN
         Der.X := 170 + ((Nb20 - 1) * 20);
         Der.Y := 63;
         Der.Type := "BIL";
         Der.Last := 20.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[13],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 20.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("vant froh"));
        DerPron := "vant froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);

      Trace := "Il appuie sur 20 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
  4 : DateStamp(DSR);
      NminEND[IndTemps] := SHORT(DSR.dsMinute);
      NticksEND[IndTemps] := SHORT(DSR.dsTick);
      Nbillets := Nbillets + 1;
      ActivEff := TRUE;
      Nb10 := Nb10 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 10.;
      IF Nb10 <= 7
      THEN
         Der.X := 170 + ((Nb10 - 1) * 18);
         Der.Y := 78;
         Der.Type := "PA";
         Der.Last := 10.;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[19],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 10.
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("dhe froh"));
        DerPron := "dhe froh";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
```

```
    Trace := "Il appuie sur 10 Frs ";
    IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
5 : DateStamp(DSR);
    NminEND[IndTemps] := SHORT(DSR.dsMinute);
    NticksEND[IndTemps] := SHORT(DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb5 := Nb5 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 5.;
    IF Nb5 <= 7
    THEN
       Der.X := 170 + ((Nb5 - 1) * 18);
       Der.Y := 97;
       Der.Type := "PA";
       Der.Last := 5.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[10],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 5.
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("senk froh"));
      DerPron := "senk froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);

    Trace := "Il appuie sur 5 Frs ";
    IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END ;
6 : DateStamp(DSR);
    NminEND[IndTemps] := SHORT(DSR.dsMinute);
    NticksEND[IndTemps] := SHORT(DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb2 := Nb2 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 2.;
    IF Nb2 <= 7
    THEN
       Der.X := 170 + ((Nb2 - 1) * 18);
       Der.Y := 113;
       Der.Type := "PA";
       Der.Last := 2.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[49],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 2.
    END;
    IF (TableauParametres[1] = 2) THEN
```

```
      Res := SayAndReturn (ADR("duh froh"));
      DerPron := "duh froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);
    Trace := "Il appuie sur 2 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END |
7 : DateStamp(DSR);
    NminEND[IndTemps] := SHORT(DSR.dsMinute);
    NticksEND[IndTemps] := SHORT(DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb1 := Nb1 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 1.;
    IF Nb1 <= 7
    THEN
       Der.X := 170 + ((Nb1 - 1) * 18);
       Der.Y := 129;
       Der.Type := "PA";
       Der.Last := 1.;
       AuMoinsUn := TRUE;
       Delay (25);
       DrawImage (Win^.RPort^,TabImages[17],Der.X,Der.Y)
    ELSE
       AuMoinsUn := FALSE;
       Der.Last := 1.
    END;
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("an froh"));
      DerPron := "an froh";
    END;
    wMove (Won^,7,1);
    wClrEndLine (Won^);
    ConvRealToString (Somme, StrSomme, 2, For);
    PutStr (Won^,ADR(StrSomme));
    wMove (Won^,15,1);
    PutStr (Won^,ADR("Frs"));
    wSetCursor (Won^,FALSE);

    Trace := "Il appuie sur 1 Frs ";
    IF TableauTraces.Indice <= 499 THEN
          CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace)
          TableauTraces.Indice :=  TableauTraces.Indice + 1;
    END |
8 : DateStamp(DSR);
    NminEND[IndTemps] := SHORT(DSR.dsMinute);
    NticksEND[IndTemps] := SHORT(DSR.dsTick);
    Nbillets := Nbillets + 1;
    ActivEff := TRUE;
    Nb05 := Nb05 + 1;
    SommePrecedente := Somme;
    Somme := Somme + 0.5;
    IF Nb05 <= 7
    THEN
       Der.X := 170 + ((Nb05 - 1) * 18);
       Der.Y := 146;
       Der.Type := "PA";
```

```
      Der.Last := 0.5;
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,TabImages[7],Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 0.5
   END;
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("sekant santeam"));
     DerPron := "sekant santeam";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 2, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);

   Trace := "Il appuie sur 0.50 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END ;
 9 : DateStamp(DSR);
   NminEND[IndTemps] := SHORT(DSR.dsMinute);
   NticksEND[IndTemps] := SHORT(DSR.dsTick);
   Nbillets := Nbillets + 1;
   ActivEff := TRUE;
   Nb02 := Nb02 + 1;
   SommePrecedente := Somme;
   Somme := Somme + 0.2;
   IF Nb02 <= 7
   THEN
      Der.X := 170 + ((Nb02 - 1) * 18);
      Der.Y := 163;
      Der.Type := "PA";
      Der.Last := 0.2;
      AuMoinsUn := TRUE;
      Delay (25);
      DrawImage (Win^.RPort^,TabImages[18],Der.X,Der.Y)
   ELSE
      AuMoinsUn := FALSE;
      Der.Last := 0.2
   END;
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("vant santeam"));
     DerPron := "vant santeam";
   END;
   wMove (Won^,7,1);
   wClrEndLine (Won^);
   ConvRealToString (Somme, StrSomme, 2, For);
   PutStr (Won^,ADR(StrSomme));
   wMove (Won^,15,1);
   PutStr (Won^,ADR("Frs"));
   wSetCursor (Won^,FALSE);

   Trace := "Il appuie sur 0.20 Frs ";
   IF TableauTraces.Indice <= 499 THEN
         CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
         TableauTraces.Indice :=  TableauTraces.Indice + 1;
   END ;
10 : DateStamp(DSR);
   NminEND[IndTemps] := SHORT(DSR.dsMinute);
   NticksEND[IndTemps] := SHORT(DSR.dsTick);
```

```
      Nbillets := Nbillets + 1;
      ActivEff := TRUE;
      Nb01 := Nb01 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.1;
      IF Nb01 <= 7
      THEN
         Der.X := 170 + ((Nb01 - 1) * 18);
         Der.Y := 180;
         Der.Type := "PA";
         Der.Last := 0.1;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[19],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 0.1
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("dhe santeam"));
        DerPron := "dhe santeam";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);

      Trace := "Il appuie sur 0.1 Frs ";
      IF TableauTraces.Indice <= 499 THEN
            CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
            TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END ;
11 :  DateStamp(DSR);
      NminEND[IndTemps] := SHORT(DSR.dsMinute);
      NticksEND[IndTemps] := SHORT(DSR.dsTick);
      Nbillets := Nbillets + 1;
      ActivEff := TRUE;
      Nb005 := Nb005 + 1;
      SommePrecedente := Somme;
      Somme := Somme + 0.05;
      IF Nb005 <= 7
      THEN
         Der.X := 170 + ((Nb005 - 1) * 18);
         Der.Y := 197;
         Der.Type := "PA";
         Der.Last := 0.05;
         AuMoinsUn := TRUE;
         Delay (25);
         DrawImage (Win^.RPort^,TabImages[10],Der.X,Der.Y)
      ELSE
         AuMoinsUn := FALSE;
         Der.Last := 0.05
      END;
      IF (TableauParametres[1] = 2) THEN
        Res := SayAndReturn (ADR("senk santeam"));
        DerPron := "senk santeam";
      END;
      wMove (Won^,7,1);
      wClrEndLine (Won^);
      ConvRealToString (Somme, StrSomme, 2, For);
      PutStr (Won^,ADR(StrSomme));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
```

```
        wSetCursor (Won^,FALSE);

        Trace := "Il appuie sur 0.05 Frs ";
        IF TableauTraces.Indice <= 499 THEN
              CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
              TableauTraces.Indice :=  TableauTraces.Indice + 1;
        END ;
    12: IF ActivEff THEN

         IF (NminEND[IndTemps] > NminBEG[IndTemps]) THEN
          NticksEND[IndTemps] := NticksEND[IndTemps] + (3000 * (NminEND[IndTem
s] - NminBEG[IndTemps]));
        END;
        ent := (NticksEND[IndTemps] - NticksBEG[IndTemps]);
        ree := real (ent);
        ree2 := (ree / 50.);
        TempsInterm[IndTemps] := entier (ree2);
        IF IndTemps < 20 THEN
          IndTemps := IndTemps + 1;
        END;
         Trace := "Il demande d'effacer ";
         IF TableauChaine.Indice <= 499
         THEN IF NOT QR
              THEN TableauChaine.Tab[TableauChaine.Indice] := 4;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                   TableauChaine.Indice := TableauChaine.Indice + 1;
              ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                        = equal)
                   THEN TableauChaine.Tab [TableauChaine.Indice] := 21;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
                        TableauChaine.Indice := TableauChaine.Indice + 1;
                   ELSE TableauChaine.Tab [TableauChaine.Indice] := 13;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
                  THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                  END;
                  ent := ActTick - AncTick;
                  ree := real(ent);
                  TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                  AncMinute := ActMinute;
                  AncTick := ActTick2;
```

```
                TableauChaine.Indice := TableauChaine.Indice + 1;
            END;
        END;
    END;
IF TableauTraces.Indice <= 499 THEN
    CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
    TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                 RelVerify});
AddGadgetTextButton (30,34, ADR("OUI"));
AddGadgetTextButton (115,34, ADR("NON"));
AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS EFFACER ?"));
GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
          GadgetMutualExcludeSet {});
GlEff := EndGadgetList ();
InitRequester (ReqEff);
WITH ReqEff DO
       OlderRequest := NIL;
       LeftEdge := 70;
       TopEdge := 8;
       Width := 180;
       Height := 50;
       ReqGadget := GlEff;
       ReqBorder := NIL;
       ReqText := NIL;
       Flags := RequesterFlagsSet {};
       BackFill := BYTE(10);
       ReqLayer := NIL;
       ImageBMap := NIL;
END;
IF Request (ReqEff, Win^) THEN
   IF (TableauParametres[1] = 2) THEN
     Res := SayAndReturn (ADR("Voolay voo effahsay ?"));
   END;
   DoneEff := FALSE;
   WHILE NOT DoneEff DO
       Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
       LOOP
          MsgEff := GetMsg (Win^.UserPort^);
          IF (MsgEff = NIL) THEN EXIT; END;
          ProcIMsg (WpEff, MsgEff);
       END;
   END;
END;
FreeGadgetList (GlEff^);
Delay(25);
IF Efface THEN
  ActivEff := FALSE;
  IF ((CompareString (Der.Type,"BIL") = equal) AND (AuMoinsUn))
    THEN DrawImage (Win^.RPort^,TabImages[24],Der.X,Der.Y);
      IF (Der.X # 170) THEN
        IF Der.Last = 100.
          THEN DrawImage (Win^.RPort^,TabImages[11],Der.X - 20,Der.
          ELSIF Der.Last = 50.
          THEN DrawImage (Win^.RPort^,TabImages[12],Der.X - 20,Der.
          ELSIF Der.Last = 20.
          THEN DrawImage (Win^.RPort^,TabImages[13],Der.X - 20,Der.
          ELSIF Der.Last = 200.
          THEN DrawImage (Win^.RPort^,TabImages[50],Der.X - 20,Der.
        END
      END
    ELSIF ((CompareString (Der.Type,"P5F") = equal) AND (AuMoinsUn)
      THEN DrawImage (Win^.RPort^,TabImages[25],Der.X,Der.Y)
```

```
            ELSIF ((CompareString (Der.Type, 'PA') = equal) AND (AuMoinsUn
              THEN DrawImage (Win^.RPort^,TabImages[26],Der.X,Der.Y)
          END;
          SommePrecedente := Somme;
          Somme := Somme - Der.Last;
          wMove (Won^,7,1);
          wClrEndLine (Won^);
          ConvRealToString (Somme, StrSomme, 2, For);
          PutStr (Won^,ADR(StrSomme));
          wMove (Won^,15,1);
          PutStr (Won^,ADR("Frs"));
          wSetCursor (Won^,FALSE);
          DerPron := "";
          IF Der.Last = 200.
           THEN  Nb200 := Nb200 - 1
           ELSIF  Der.Last = 100.
           THEN  Nb100 := Nb100 - 1
           ELSIF  Der.Last = 50.
           THEN  Nb50 := Nb50 - 1
           ELSIF  Der.Last = 20.
           THEN  Nb20 := Nb20 - 1
           ELSIF  Der.Last = 10.
           THEN  Nb10 := Nb10 - 1
           ELSIF  Der.Last = 5.
           THEN  Nb5 := Nb5 - 1
           ELSIF  Der.Last = 2.
           THEN  Nb2 := Nb2 - 1
           ELSIF  Der.Last = 1.
           THEN  Nb1 := Nb1 - 1
           ELSIF  Der.Last = 0.5
           THEN  Nb05 := Nb05 - 1
           ELSIF  Der.Last = 0.2
           THEN  Nb02 := Nb02 - 1
           ELSIF  Der.Last = 0.1
           THEN  Nb01 := Nb01 - 1
           ELSIF Der.Last = 0.05
           THEN  Nb005 := Nb005 - 1
           ELSE
          END;
        END;
        DateStamp(DSR);
        NminBEG[IndTemps] := SHORT(DSR.dsMinute);
        NminEND[IndTemps] := NminBEG[IndTemps];
        NticksBEG[IndTemps] := SHORT(DSR.dsTick);
        NticksEND[IndTemps] := NticksBEG[IndTemps];
      END ;
13 :  IF (NminEND[IndTemps] > NminBEG[IndTemps]) THEN
        NticksEND[IndTemps] := NticksEND[IndTemps] +
          (3000 * (NminEND[IndTemps] - NminBEG[IndTemps]));
       END;
       ent := (NticksEND[IndTemps] - NticksBEG[IndTemps]);
       ree := real (ent);
       ree2 := (ree / 50.);
       TempsInterm[IndTemps] := entier (ree2);
       IF IndTemps < 20 THEN
         IndTemps := IndTemps + 1;
       END;
        Trace := "Il appuie sur ok ";
        IF TableauChaine.Indice <= 499
        THEN IF NOT QR
             THEN TableauChaine.Tab[TableauChaine.Indice] := 6;
                  DateStamp(DSR);
                  ActMinute := SHORT(DSR.dsMinute);
                  ActTick := SHORT(DSR.dsTick);
                  ActTick2 := ActTick;
                  IF ActMinute > AncMinute
```

```
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
             TableauChaine.Indice := TableauChaine.Indice + 1;
        ELSE IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                  = equal)
             THEN TableauChaine.Tab [TableauChaine.Indice] := 23;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
                  TableauChaine.Indice := TableauChaine.Indice + 1;
             ELSE TableauChaine.Tab [TableauChaine.Indice] := 15;
            DateStamp(DSR);
            ActMinute := SHORT(DSR.dsMinute);
            ActTick := SHORT(DSR.dsTick);
            ActTick2 := ActTick;
            IF ActMinute > AncMinute
            THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
            END;
            ent := ActTick - AncTick;
            ree := real(ent);
            TableauTemps [TableauChaine.Indice] := entier(ree/50.);
            AncMinute := ActMinute;
            AncTick := ActTick2;
                  TableauChaine.Indice := TableauChaine.Indice + 1;
             END;
        END;
   END;
IF TableauTraces.Indice <= 499 THEN
       CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
       TableauTraces.Indice :=  TableauTraces.Indice + 1;
END;
BeginGadgetList ();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                  RelVerify});
AddGadgetTextButton (30,34, ADR("OUI"));
AddGadgetTextButton (115,34, ADR("NON"));
AddGadgetTextButton (20,9, ADR("AVEZ-VOUS FINI ?"));
GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
           GadgetMutualExcludeSet {});
G1Fin := EndGadgetList ();
InitRequester (ReqFin);
WITH ReqFin DO
       OlderRequest := NIL;
       LeftEdge := 70;
       TopEdge := 8;
       Width := 180;
       Height := 50;
       ReqGadget := G1Fin;
       ReqBorder := NIL;
       ReqText := NIL;
       Flags := RequesterFlagsSet {};
```

```
            BackFill := BYTE(10);
            ReqLayer := NIL;
            ImageBMap := NIL;
      END;
      IF Request (ReqFin, Win^) THEN
         IF (TableauParametres[1] = 2) THEN
            Res := SayAndReturn (ADR("ahvay voo feeny ?"));
         END;
         DoneFin := FALSE;
         WHILE NOT DoneFin DO
            Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
            LOOP
               MsgFin := GetMsg (Win^.UserPort^);
               IF (MsgFin = NIL) THEN EXIT; END;
               ProcIMsg (WpFin, MsgFin);
            END;
         END;
      END;
      FreeGadgetList (GlFin^);
      Delay(25);
      IF Fini THEN Done := TRUE
      ELSE
        DateStamp(DSR);
        NminBEG[IndTemps] := SHORT(DSR.dsMinute);
        NminEND[IndTemps] := NminBEG[IndTemps];
        NticksBEG[IndTemps] := SHORT(DSR.dsTick);
        NticksEND[IndTemps] := NticksBEG[IndTemps];

      END;
14 :IF QR = TRUE
     THEN
       IF (NminEND[IndTemps] > NminBEG[IndTemps]) THEN
        NticksEND[IndTemps] := NticksEND[IndTemps] +
                      (3000 * (NminEND[IndTemps] - NminBEG[IndTemps]));
       END;
       ent := (NticksEND[IndTemps] - NticksBEG[IndTemps]);
       ree := real (ent);
       ree2 := (ree / 50.);
       TempsInterm[IndTemps] := entier (ree2);
       IF IndTemps < 20 THEN
         IndTemps := IndTemps + 1;
       END;
       Trace := "Il appuie sur QUITTER ";
       IF TableauChaine.Indice <= 499
       THEN IF (CompareString (PPron,"Cobya ah vay voo daypensay ?")
                          = equal)
            THEN TableauChaine.Tab [TableauChaine.Indice] := 22;
                 DateStamp(DSR);
                 ActMinute := SHORT(DSR.dsMinute);
                 ActTick := SHORT(DSR.dsTick);
                 ActTick2 := ActTick;
                 IF ActMinute > AncMinute
                 THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                 END;
                 ent := ActTick - AncTick;
                 ree := real(ent);
                 TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                 AncMinute := ActMinute;
                 AncTick := ActTick2;
                  TableauChaine.Indice := TableauChaine.Indice + 1;
            ELSE TableauChaine.Tab [TableauChaine.Indice] := 14;
                 DateStamp(DSR);
                 ActMinute := SHORT(DSR.dsMinute);
                 ActTick := SHORT(DSR.dsTick);
                 ActTick2 := ActTick;
                 IF ActMinute > AncMinute
```

```
                THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
                END;
                ent := ActTick - AncTick;
                ree := real(ent);
                TableauTemps [TableauChaine.Indice] := entier(ree/50.);
                AncMinute := ActMinute;
                AncTick := ActTick2;
                 TableauChaine.Indice := TableauChaine.Indice + 1;
           END;
       END;
      IF TableauTraces.Indice <= 499 THEN
           CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
           TableauTraces.Indice :=  TableauTraces.Indice + 1;
      END;
      BeginGadgetList ();
      GadgetTypeReq := TRUE;
      GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                       RelVerify});
      AddGadgetTextButton (30,34, ADR("OUI"));
      AddGadgetTextButton (115,34, ADR("NON"));
      AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS QUITTER ?"));
      GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
               GadgetMutualExcludeSet {});
      GlQui := EndGadgetList ();
      InitRequester (ReqQui);
      WITH ReqQui DO
           OlderRequest := NIL;
           LeftEdge := 70;
           TopEdge := 8;
           Width := 180;
           Height := 50;
           ReqGadget := GlQui;
           ReqBorder := NIL;
           ReqText := NIL;
           Flags := RequesterFlagsSet {};
           BackFill := BYTE(10);
           ReqLayer := NIL;
           ImageBMap := NIL;
      END;
      IF Request (ReqQui, Win^) THEN
         IF (TableauParametres[1] = 2) THEN
           Res := SayAndReturn (ADR("Voolay voo kittay ?"));
         END;
         DoneQui := FALSE;
         WHILE NOT DoneQui DO
           Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
           LOOP
              MsgQui := GetMsg (Win^.UserPort^);
              IF (MsgQui = NIL) THEN EXIT; END;
              ProcIMsg (WpQui, MsgQui);
           END;
       END;
      END;
      FreeGadgetList (GlQui^);
      Delay(25);
      IF Quitter THEN Done := TRUE;ES := "quitter";
      ELSE
        DateStamp(DSR);
        NminBEG[IndTemps] := SHORT(DSR.dsMinute);
        NminEND[IndTemps] := NminBEG[IndTemps];
        NticksBEG[IndTemps] := SHORT(DSR.dsTick);
        NticksEND[IndTemps] := NticksBEG[IndTemps];
      END;
    ELSE
      IF (NminEND[IndTemps] > NminBEG[IndTemps]) THEN
       NticksEND[IndTemps] := NticksEND[IndTemps] +
```

```
                        (3000 * (NminEND[IndTemps] - NminBEG[IndTemps]))
END;
ent := (NticksEND[IndTemps] - NticksBEG[IndTemps]);
ree := real (ent);
ree2 := (ree / 50.);
TempsInterm[IndTemps] := entier (ree2);
IF IndTemps < 20 THEN
  IndTemps := IndTemps + 1;
END;

 IF TableauChaine.Indice <= 499
 THEN TableauChaine.Tab[TableauChaine.Indice] := 5;
          DateStamp(DSR);
          ActMinute := SHORT(DSR.dsMinute);
          ActTick := SHORT(DSR.dsTick);
          ActTick2 := ActTick;
          IF ActMinute > AncMinute
          THEN ActTick := ActTick + (3000 * (ActMinute-AncMinute));
          END;
          ent := ActTick - AncTick;
          ree := real(ent);
          TableauTemps [TableauChaine.Indice] := entier(ree/50.);
          AncMinute := ActMinute;
          AncTick := ActTick2;
      TableauChaine.Indice := TableauChaine.Indice + 1;
 END;
 Trace := "Il appuie sur RECOMMENCER ";
 IF TableauTraces.Indice <= 499 THEN
      CopyString ( TableauTraces.Tab [TableauTraces.Indice] ,Trace);
      TableauTraces.Indice :=  TableauTraces.Indice + 1;
 END;
 BeginGadgetList ();
 GadgetTypeReq := TRUE;
 GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                  RelVerify});
 AddGadgetTextButton (30,34, ADR("OUI"));
 AddGadgetTextButton (115,34, ADR("NON"));
 AddGadgetTextButton (2,9, ADR("VOULEZ-VOUS RECOMMENCER ?"));
 GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
          GadgetMutualExcludeSet {});
 GlRec := EndGadgetList ();
 InitRequester (ReqRec);
 WITH ReqRec DO
      OlderRequest := NIL;
      LeftEdge := 50;
      TopEdge := 8;
      Width := 220;
      Height := 50;
      ReqGadget := GlRec;
      ReqBorder := NIL;
      ReqText := NIL;
      Flags := RequesterFlagsSet {};
      BackFill := BYTE(10);
      ReqLayer := NIL;
      ImageBMap := NIL;
 END;
 IF Request (ReqRec, Win^) THEN
    IF (TableauParametres[1] = 2) THEN
      Res := SayAndReturn (ADR("Vooley voo ra commohsay ?"));
    END;
    DoneRec := FALSE;
    WHILE NOT DoneRec DO
      Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
      LOOP
         MsgRec := GetMsg (Win^.UserPort^);
         IF (MsgRec = NIL) THEN EXIT; END;
```

```
            ProcIMsg (WpRec, MsgRec);
        END;
    END;
END;
FreeGadgetList (GlRec^);
Delay(25);
IF Recom THEN
        DrawImage (Win^.RPort^,TabImages[24],170,15);
        DrawImage (Win^.RPort^,TabImages[24],190,15);
        DrawImage (Win^.RPort^,TabImages[24],210,15);
        DrawImage (Win^.RPort^,TabImages[24],230,15);
        DrawImage (Win^.RPort^,TabImages[24],250,15);
        DrawImage (Win^.RPort^,TabImages[24],170,31);
        DrawImage (Win^.RPort^,TabImages[24],190,31);
        DrawImage (Win^.RPort^,TabImages[24],210,31);
        DrawImage (Win^.RPort^,TabImages[24],230,31);
        DrawImage (Win^.RPort^,TabImages[24],250,31);
        DrawImage (Win^.RPort^,TabImages[24],170,47);
        DrawImage (Win^.RPort^,TabImages[24],190,47);
        DrawImage (Win^.RPort^,TabImages[24],210,47);
        DrawImage (Win^.RPort^,TabImages[24],230,47);
        DrawImage (Win^.RPort^,TabImages[24],250,47);
        DrawImage (Win^.RPort^,TabImages[24],170,63);
        DrawImage (Win^.RPort^,TabImages[24],190,63);
        DrawImage (Win^.RPort^,TabImages[24],210,63);
        DrawImage (Win^.RPort^,TabImages[24],230,63);
        DrawImage (Win^.RPort^,TabImages[24],250,63);
        DrawImage (Win^.RPort^,TabImages[26],170,78);
        DrawImage (Win^.RPort^,TabImages[26],188,78);
        DrawImage (Win^.RPort^,TabImages[26],206,78);
        DrawImage (Win^.RPort^,TabImages[26],224,78);
        DrawImage (Win^.RPort^,TabImages[26],242,78);
        DrawImage (Win^.RPort^,TabImages[26],260,78);
        DrawImage (Win^.RPort^,TabImages[26],278,78);
        DrawImage (Win^.RPort^,TabImages[26],170,97);
        DrawImage (Win^.RPort^,TabImages[26],188,97);
        DrawImage (Win^.RPort^,TabImages[26],206,97);
        DrawImage (Win^.RPort^,TabImages[26],224,97);
        DrawImage (Win^.RPort^,TabImages[26],242,97);
        DrawImage (Win^.RPort^,TabImages[26],260,97);
        DrawImage (Win^.RPort^,TabImages[26],278,97);
        DrawImage (Win^.RPort^,TabImages[26],170,113);
        DrawImage (Win^.RPort^,TabImages[26],188,113);
        DrawImage (Win^.RPort^,TabImages[26],206,113);
        DrawImage (Win^.RPort^,TabImages[26],224,113);
        DrawImage (Win^.RPort^,TabImages[26],242,113);
        DrawImage (Win^.RPort^,TabImages[26],260,113);
        DrawImage (Win^.RPort^,TabImages[26],278,113);
        DrawImage (Win^.RPort^,TabImages[26],170,129);
        DrawImage (Win^.RPort^,TabImages[26],188,129);
        DrawImage (Win^.RPort^,TabImages[26],206,129);
        DrawImage (Win^.RPort^,TabImages[26],224,129);
        DrawImage (Win^.RPort^,TabImages[26],242,129);
        DrawImage (Win^.RPort^,TabImages[26],260,129);
        DrawImage (Win^.RPort^,TabImages[26],278,129);
        DrawImage (Win^.RPort^,TabImages[26],170,146);
        DrawImage (Win^.RPort^,TabImages[26],188,146);
        DrawImage (Win^.RPort^,TabImages[26],206,146);
        DrawImage (Win^.RPort^,TabImages[26],224,146);
        DrawImage (Win^.RPort^,TabImages[26],242,146);
        DrawImage (Win^.RPort^,TabImages[26],260,146);
        DrawImage (Win^.RPort^,TabImages[26],278,146);
        DrawImage (Win^.RPort^,TabImages[26],170,163);
        DrawImage (Win^.RPort^,TabImages[26],188,163);
        DrawImage (Win^.RPort^,TabImages[26],206,163);
        DrawImage (Win^.RPort^,TabImages[26],224,163);
```

```
                DrawImage (Win^.RPort^,TabImages[26],242,163);
                DrawImage (Win^.RPort^,TabImages[26],260,163);
                DrawImage (Win^.RPort^,TabImages[26],278,163);
                DrawImage (Win^.RPort^,TabImages[26],170,180);
                DrawImage (Win^.RPort^,TabImages[26],188,180);
                DrawImage (Win^.RPort^,TabImages[26],206,180);
                DrawImage (Win^.RPort^,TabImages[26],224,180);
                DrawImage (Win^.RPort^,TabImages[26],242,180);
                DrawImage (Win^.RPort^,TabImages[26],260,180);
                DrawImage (Win^.RPort^,TabImages[26],278,180);
                DrawImage (Win^.RPort^,TabImages[26],170,197);
                DrawImage (Win^.RPort^,TabImages[26],188,197);
                DrawImage (Win^.RPort^,TabImages[26],206,197);
                DrawImage (Win^.RPort^,TabImages[26],224,197);
                DrawImage (Win^.RPort^,TabImages[26],242,197);
                DrawImage (Win^.RPort^,TabImages[26],260,197);
                DrawImage (Win^.RPort^,TabImages[26],278,197);

                Somme := 0.;
                wMove (Won^,7,1);
                wClrEndLine (Won^);
                ConvRealToString (Somme, StrSomme, 2, For);
                PutStr (Won^,ADR(StrSomme));
                wMove (Won^,15,1);
                PutStr (Won^,ADR("Frs"));
                wSetCursor (Won^,FALSE);
                ES := "";
                SommeMax := Max;
                Nb200 := 0;
                Nb100 := 0;
                Nb50 := 0;
                Nb20 := 0;
                Nb10 := 0;
                Nb5 := 0;
                Nb2 := 0;
                Nb1 := 0;
                Nb05 := 0;
                Nb02 := 0;
                Nb01 := 0;
                Nb005 := 0;
                Der.Last := 0.;
                Der.X := 150;
                AuMoinsUn := FALSE;
                ActivEff := FALSE;
                For := Decimal;
                CopyString (DerPron,PPron);
              END;
            DateStamp(DSR);
            NminBEG[IndTemps] := SHORT(DSR.dsMinute);
            NminEND[IndTemps] := NminBEG[IndTemps];
            NticksBEG[IndTemps] := SHORT(DSR.dsTick);
            NticksEND[IndTemps] := NticksBEG[IndTemps];

             END|
       15:  IF (TableauParametres[1] = 2) THEN
               Res := SayAndReturn (ADR(DerPron));
            END;
    END;
CloseSpeech ();
END;
END GadgetHandlerBS;


PROCEDURE SaisieSommeBillets (VAR Phrase : ARRAY OF CHAR;
                              VAR PhrasePron : ARRAY OF CHAR;
                              Quit :BOOLEAN; VAR Valeur : REAL;
```

```
                                VAR EtatSortie : Str20);

BEGIN
    CopyString (PPron,PhrasePron);
    ES := "";
    QR := Quit;
    Somme := 0.;
    SommeMax := Max;
    Nb200 := 0;
    Nb100 := 0;
    Nb50 := 0;
    Nb20 := 0;
    Nb10 := 0;
    Nb5 := 0;
    Nb2 := 0;
    Nb1 := 0;
    Nb05 := 0;
    Nb02 := 0;
    Nb01 := 0;
    Nb005 := 0;
    Der.Last := 0.;
    Der.X := 150;
    AuMoinsUn := FALSE;
    ActivEff := FALSE;
    For := Decimal;
    Nbillets := 0;
    IndTemps := 1;

    WITH Wp DO
      procGadgetUp := GadgetHandlerBS;
    END;
    WITH WpEff DO
      procGadgetUp := GadgetHandlerEffBS;
    END;
    WITH WpQui DO
      procGadgetUp := GadgetHandlerQuiBS;
    END;
    WITH WpFin DO
      procGadgetUp := GadgetHandlerFinBS;
    END;
    WITH WpRec DO
      procGadgetUp := GadgetHandlerRecBS;
    END;
    Scr := CreateScreen (320,230,5,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 0FCAH;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
```

```
cmap[21] := 0444H;
cmap[22] := 0555H;
cmap[23] := 0666H;
cmap[24] := 0777H;
cmap[25] := 0888H;
cmap[26] := 0999H;
cmap[27] := 0AAAH;
cmap[28] := 0CCCH;
cmap[29] := 0DDDH;
cmap[30] := 0EEEH;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

ImgPtr := FindImageTable(33333333H, ImgCount);
IF (ImgPtr # NIL) AND (ImgCount # 0) THEN
  Indice := 0;
  WHILE (Indice <= ImgCount-1) DO
    WITH TabImages[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr^[Indice].Width;
      Height      := ImgPtr^[Indice].Height;
      Depth       := ImgPtr^[Indice].Depth;
      ImageData   := ImgPtr^[Indice].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;
END;
ImgPtr2 := FindImageTable(44444444H, ImgCount2);
 IF (ImgPtr2 # NIL) AND (ImgCount2 # 0) THEN
  WHILE (Indice-ImgCount <= ImgCount2-1) DO
    WITH TabImages[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr2^[Indice-ImgCount].Width;
      Height      := ImgPtr2^[Indice-ImgCount].Height;
      Depth       := ImgPtr2^[Indice-ImgCount].Depth;
      ImageData   := ImgPtr2^[Indice-ImgCount].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;
 END;
 ImgPtr3 := FindImageTable(77777777H, ImgCount3);
 IF (ImgPtr3 # NIL) AND (ImgCount3 # 0) THEN
    WITH TabImages[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr3^[9].Width;
      Height      := ImgPtr3^[9].Height;
      Depth       := ImgPtr3^[9].Depth;
      ImageData   := ImgPtr3^[9].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
 END;
 Indice := Indice + 1;

 ImgPtr4 := FindImageTable(66666666H, ImgCount4);
 IF (ImgPtr4 # NIL) AND (ImgCount4 # 0) THEN
```

```
    list6 := 3;
    WHILE (list6 <= ImgCount4-1) DO
      WITH TabImages[Indice] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr4^[list6].Width;
        Height      := ImgPtr4^[list6].Height;
        Depth       := ImgPtr4^[list6].Depth;
        ImageData   := ImgPtr4^[list6].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
      END;
      list6 := list6 + 1;
      Indice := Indice + 1;
    END;
END;
ImgPtr5 := FindImageTable(11111114H, ImgCount5);
 IF (ImgPtr5 # NIL) AND (ImgCount5 # 0) THEN
  list7 := 3;
  WHILE (list7 <= ImgCount5-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr5^[list7].Width;
       Height      := ImgPtr5^[list7].Height;
       Depth       := ImgPtr5^[list7].Depth;
       ImageData   := ImgPtr5^[list7].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list7 := list7 + 1;
     Indice := Indice + 1;
   END;
END;
ImgPtr6 := FindImageTable(22222223H, ImgCount6);
 IF (ImgPtr6 # NIL) AND (ImgCount6 # 0) THEN
  list7 := 0;
  Indice := 46;
  WHILE (list7 <= ImgCount6-1) DO
     WITH TabImages[Indice] DO
       LeftEdge    := 0;
       TopEdge     := 0;
       Width       := ImgPtr6^[list7].Width;
       Height      := ImgPtr6^[list7].Height;
       Depth       := ImgPtr6^[list7].Depth;
       ImageData   := ImgPtr6^[list7].Data;
       PlanePick   := BYTE(31);
       PlaneOnOff  := BYTE(31);
       NextImage := NIL;
     END;
     list7 := list7 + 1;
     Indice := Indice + 1;
   END;
END;
    BeginGadgetList();
    AddGadgetImageButton (10,10,TabImages[48]);
    AddGadgetImageButton (10,47,TabImages[0]);
    AddGadgetImageButton (10,84,TabImages[1]);
    AddGadgetImageButton (10,121,TabImages[2]);
    AddGadgetImageButton (88,5,TabImages[47]);
    AddGadgetImageButton (85,34,TabImages[4]);
    AddGadgetImageButton (87,70,TabImages[46]);
    AddGadgetImageButton (90,101,TabImages[6]);
    AddGadgetImageButton (92,125,TabImages[7]);
```

```
AddGadgetImageButton (91,143,TabImages[8]);
AddGadgetImageButton (92,164,TabImages[9]);
AddGadgetImageButton (93,183,TabImages[10]);
AddGadgetImageButton (4,200,TabImages[20]);
AddGadgetImageButton (121,200,TabImages[21]);
IF Quit = TRUE THEN AddGadgetImageButton (63,200,TabImages[22])
               ELSE AddGadgetImageButton (63,200,TabImages[36]) END;
    IF TableauParametres[1] = 2 THEN
AddGadgetImageButton (6,160,TabImages[35]);
END;
Gl := EndGadgetList();
IF (Gl # NIL) THEN
   Win := CreateWindow (0,0,320,230,WIDCMP,WFlags,Gl,Scr,NIL);
   Win2 := CreateWindow (134,72,23,102,WIDCMP,WFlags2,NIL,Scr,NIL);

   Won := CreateWindow (168,212,152,15,WIDCMP,WFlags,NIL,Scr,NIL);
   IF (Win # NIL) AND (Won # NIL)  THEN

    IF CreateConsole (Won^) THEN
      wClrScr (Won^);
      PutStr (Won^,ADR("Total"));
      wMove (Won^,15,1);
      PutStr (Won^,ADR("Frs"));
      wSetCursor (Won^,FALSE);
      DrawImage (Win^.RPort^,TabImages[28],165,0);
      DrawImage (Win^.RPort^,TabImages[28],165,100);
      IF NOT QR THEN
        DrawImage (Win^.RPort^,TabImages[43],0,0);
        DrawImage (Win^.RPort^,TabImages[43],130,0);
        DrawImage (Win^.RPort^,TabImages[43],190,0);
        DrawImage (Win^.RPort^,TabImages[43],0,227);
        DrawImage (Win^.RPort^,TabImages[43],130,227);
        DrawImage (Win^.RPort^,TabImages[43],190,227);
        DrawImage (Win^.RPort^,TabImages[45],0,0);
        DrawImage (Win^.RPort^,TabImages[45],0,100);
        DrawImage (Win^.RPort^,TabImages[45],317,0);
        DrawImage (Win^.RPort^,TabImages[45],317,100);
        DrawImage (Win^.RPort^,TabImages[42],123,3);
        DrawImage (Win^.RPort^,TabImages[45],165,0);
        DrawImage (Win^.RPort^,TabImages[45],165,100);
        DrawImage (Win^.RPort^,TabImages[39],175,5);

      END;
      IF (CompareString (PhrasePron,"Cobya ah vay voo ra su darjo ?")
              = equal)
      THEN
        DrawImage (Win^.RPort^,TabImages[31],0,0);
        DrawImage (Win^.RPort^,TabImages[31],130,0);
        DrawImage (Win^.RPort^,TabImages[31],190,0);
        DrawImage (Win^.RPort^,TabImages[31],0,227);
        DrawImage (Win^.RPort^,TabImages[31],130,227);
        DrawImage (Win^.RPort^,TabImages[31],190,227);
        DrawImage (Win^.RPort^,TabImages[29],0,0);
        DrawImage (Win^.RPort^,TabImages[29],0,100);
        DrawImage (Win^.RPort^,TabImages[29],317,0);
        DrawImage (Win^.RPort^,TabImages[29],317,100);
        DrawImage (Win^.RPort^,TabImages[33],123,3);
        DrawImage (Win^.RPort^,TabImages[29],165,0);
        DrawImage (Win^.RPort^,TabImages[29],165,100);
        DrawImage (Win^.RPort^,TabImages[41],175,5);

      ELSIF (CompareString (PhrasePron,"Cobya ah vay voo daypensay ?'
              = equal)
      THEN
        DrawImage (Win^.RPort^,TabImages[32],0,0);
        DrawImage (Win^.RPort^,TabImages[32],130,0);
```

```
                DrawImage (Win^.RPort^,TabImages[32],190,0);
                DrawImage (Win^.RPort^,TabImages[32],0,227);
                DrawImage (Win^.RPort^,TabImages[32],130,227);
                DrawImage (Win^.RPort^,TabImages[32],190,227);
                DrawImage (Win^.RPort^,TabImages[30],0,0);
                DrawImage (Win^.RPort^,TabImages[30],0,100);
                DrawImage (Win^.RPort^,TabImages[30],317,0);
                DrawImage (Win^.RPort^,TabImages[30],317,100);
                DrawImage (Win^.RPort^,TabImages[34],123,3);
                DrawImage (Win^.RPort^,TabImages[30],165,0);
                DrawImage (Win^.RPort^,TabImages[30],165,100);
                DrawImage (Win^.RPort^,TabImages[40],175,5);

              END;
              IF TableauParametres[1] = 2 THEN
                IF OpenSpeech () THEN
                        Res := SayAndReturn (ADR(PhrasePron));
                        CopyString (DerPron,PhrasePron);
                        CloseSpeech ()
                END;
              END;

              SA := Sactuel;
              DateStamp (DSR);
              NminBEG[IndTemps] := SHORT (DSR.dsMinute);
              NticksBEG[IndTemps] := SHORT (DSR.dsTick);
              Done := FALSE;
              WHILE (NOT Done) DO
                Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
                LOOP
                  Msg := GetMsg(Win^.UserPort^);
                  IF (Msg = NIL) THEN EXIT; END;
                  ProcIMsg (Wp, Msg);
                END;
              END;
              IndTemps := IndTemps - 1;
              IF (( Nbillets > 0) AND (NOT QR) )
              THEN
                 II := 1;
                 WHILE (II <= IndTemps) DO
                    TempsSIBillets := TempsSIBillets + TempsInterm[II];
                    II := II + 1;
                 END;
                 ree := real(TempsSIBillets);
                 ree2 := real (Nbillets);
                 TempsSIBillets := entier (ree / ree2);
              END;

              DeleteConsole (Won^);
              ELSE WriteString ("Console non créée")
            END;
          EtatSortie := ES;
          Valeur := Somme;
          CloseWindow(Won^);
          CloseWindow(Win^);
          END;
          FreeGadgetList (Gl^);
        END;
      CloseScreen(Scr^);
    END;
IF Quit = TRUE
 THEN
  TabNombre[0] := Nb200;
  TabNombre[1] := Nb100;
  TabNombre[2] := Nb50;
  TabNombre[3] := Nb20;
```

```
  TabNombre[4] := Nb10;
  TabNombre[5] := Nb5;
  TabNombre[6] := Nb2;
  TabNombre[7] := Nb1;
  TabNombre[8] := Nb05;
  TabNombre[9] := Nb02;
  TabNombre[10] := Nb01;
  TabNombre[11] := Nb005;
 ELSE
  TabNombreInit[0] := Nb200;
  TabNombreInit[1] := Nb100;
  TabNombreInit[2] := Nb50;
  TabNombreInit[3] := Nb20;
  TabNombreInit[4] := Nb10;
  TabNombreInit[5] := Nb5;
  TabNombreInit[6] := Nb2;
  TabNombreInit[7] := Nb1;
  TabNombreInit[8] := Nb05;
  TabNombreInit[9] := Nb02;
  TabNombreInit[10] := Nb01;
  TabNombreInit[11] := Nb005;
END;

END SaisieSommeBillets;

(*  VAR

  p,pp : ARRAY [0..40] OF CHAR;
  q : BOOLEAN;
  v : REAL;
  e : Str20;

BEGIN

  p := "Combien avez-vous d'argent ?";
  pp := "Cobya ah vay voo darjo ?";
  q := FALSE;
  SaisieSommeBillets (p,pp,q,v,e);    *)

END SSBillets.
```

```
IMPLEMENTATION MODULE SRBillets;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT BYTE, ADR, SHORT;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,IntuiText,IntuiTextPtr, BorderPtr;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleSpeech IMPORT OpenSpeech,CloseSpeech,SayAndReturn;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLin
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation,ConcatString,
        StringLength,LocateChar,ExtractSubString;
FROM VariablesGlobales IMPORT Nom, Str3, Str20,Str40,Str100,TabNombreInit,
                              TableauTraces,TRecettes,TDepenses,
                              TableauParametres,Sinit,Max,TabNombre,Sactuel;
FROM Utilitaires IMPORT ConvPrononcable,RegletteJour,RegHeure;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM FindImages IMPORT ImageDescTablePtr, FindImageTable;
FROM Rasters IMPORT SetRast;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString, RealToStringFormat,
                              ConvStringToReal;


CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Activate,Borderless};
  NomBlanc = "                    ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
              Last : REAL;
            END;

VAR
  Done,DoneOK : BOOLEAN;
  Efface,Quitter,Fini,DoneEff,DoneQui,DoneFin,AuMoinsUn,ActivEff : BOOLEAN;
  Gl,GlEff,GlQui,GlFin,GlOK : GadgetPtr;
  Wp,WpEff,WpQui,WpFin,WpOK : WindowProc;
  Req,ReqEff,ReqQui,ReqFin,ReqOK : Requester;
  Sig : SignalSet;
  Msg, MsgEff, MsgQui, MsgFin,MsgOK : IntuiMessagePtr;
  Nb5000,Nb1000,Nb500,Nb100,Nb50bis,Nb50,Nb20,Nb10,Nb5,Nb2,Nb1,Nb05,
  Nb02,Nb01,Nb005 : CARDINAL;
```

```
  Confirmation,Trace : Str40;
  Der : Dernier;
  Somme, SommePrecedente, SommeMax,ValeurMax,ValeurPrec : REAL;
  StrSomme,ES,fonc,SommeString,Maxaf : Str20;
  TePtr : IntuiTextPtr;
  For : RealToStringFormat;
  Win,Won,Win1,Win2,Win3,Win4,Win5,Win6,Win61,Wun,Wun2 : WindowPtr;
  Scr : ScreenPtr;
  cmap : ARRAY [0..31] OF CARDINAL;
  TabImages : ARRAY [0..29] OF Image;
  ImgPtr,ImgPtr2,ImgPtr3 : ImageDescTablePtr;
  ImgCount,ImgCount2,ImgCount3,Indice,i,I,X,Y : CARDINAL;
  Res : BOOLEAN;
  SomPron,PhrasePron : Str100;
  Phrase  : Str40;
  Phrase2,Diff : ARRAY [0..39] OF CHAR;
  E : INTEGER;

PROCEDURE GadgetHandler (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN

  Done := TRUE;
END GadgetHandler;

PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : DoneOK := TRUE |
     1 :
  END;
END GadgetHandlerOK;

PROCEDURE OKRequester(EPMReq : INTEGER);
BEGIN


  WITH WpOK DO
      procGadgetUp := GadgetHandlerOK;
  END;

          IF OpenSpeech () THEN
          BeginGadgetList ();
          GadgetTypeReq := TRUE;
          GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                           RelVerify});
          AddGadgetTextButton (125,40, ADR(" OK "));
          IF (EPMReq = 1)
           THEN ConvRealToString (Sinit - Sactuel,Diff,2,Decimal);
                CopyString (Phrase2,"Il y a une erreur de ");
                ConcatString (Phrase2,Diff);
                ConcatString (Phrase2," Frs.")
           ELSE ConvRealToString (Sactuel - Sinit,Diff,2,Decimal);
                CopyString (Phrase2,"Il y a une erreur de ");
                ConcatString (Phrase2,Diff);
                ConcatString (Phrase2," Frs.")

          END;
          AddGadgetTextButton (2,15, ADR(Phrase2));
          GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                     GadgetMutualExcludeSet {});
          G1OK := EndGadgetList ();
          InitRequester (ReqOK);
          WITH ReqOK DO
                 OlderRequest := NIL;
                 LeftEdge := 3;
                 TopEdge := 65;
```

```
                Width := 268;
                Height := 60;
                ReqGadget := GlOK;
                ReqBorder := NIL;
                ReqText := NIL;
                Flags := RequesterFlagsSet {};
                BackFill := BYTE(10);
                ReqLayer := NIL;
                ImageBMap := NIL;
            END;
            IF Request (ReqOK, Win1^) THEN
            IF (EPMReq = 1) THEN
              IF (TableauParametres[1] = 2)
                THEN Res := SayAndReturn (ADR("Il y a une erreur"));
              END;
            ELSE
              IF (TableauParametres[1] = 2)
                THEN Res := SayAndReturn (ADR("Il y a une erreur"));
              END;
            END;
            DoneOK := FALSE;
            WHILE NOT DoneOK DO
              Sig := Wait(SignalSet{CARDINAL(Win1^.UserPort^.mpSigBit)});
              LOOP
                MsgOK := GetMsg (Win1^.UserPort^);
                IF (MsgOK = NIL) THEN EXIT; END;
                ProcIMsg (WpOK, MsgOK);
              END;
            END;
            END;
             FreeGadgetList (GlOK^);
CloseSpeech();
END;
END OKRequester;

PROCEDURE SommeRestanteBillets (EPMReq : INTEGER);

BEGIN

 IF OpenSpeech () THEN END;
  ValeurMax := Max;

  WITH Wp DO
      procGadgetUp := GadgetHandler;
  END;
  Scr := CreateScreen (320,240,5,NIL);
  IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0E00H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
```

```
cmap[19] := 0FCAH;
cmap[20] := 0333H;
cmap[21] := 0444H;
cmap[22] := 0555H;
cmap[23] := 0666H;
cmap[24] := 0777H;
cmap[25] := 0888H;
cmap[26] := 0999H;
cmap[27] := 0AAAH;
cmap[28] := 0CCCH;
cmap[29] := 0DDDH;
cmap[30] := 0E00H;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),32);

ImgPtr := FindImageTable(22222223H, ImgCount);
  Indice := 0;
    WITH TabImages[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr^[4].Width;
      Height      := ImgPtr^[4].Height;
      Depth       := ImgPtr^[4].Depth;
      ImageData   := ImgPtr^[4].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;

ImgPtr2 := FindImageTable(33333333H, ImgCount2);
  WHILE (Indice <= 3) DO
    WITH TabImages[Indice] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr2^[Indice+10].Width;
      Height      := ImgPtr2^[Indice+10].Height;
      Depth       := ImgPtr2^[Indice+10].Depth;
      ImageData   := ImgPtr2^[Indice+10].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
    END;
    Indice := Indice + 1;
  END;
ImgPtr3 := FindImageTable(44444444H, ImgCount3);
WITH TabImages[4] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr3^[5].Width;
      Height      := ImgPtr3^[5].Height;
      Depth       := ImgPtr3^[5].Depth;
      ImageData   := ImgPtr3^[5].Data;
      PlanePick   := BYTE(31);
      PlaneOnOff  := BYTE(31);
      NextImage := NIL;
END;

WITH TabImages[5] DO
      LeftEdge    := 0;
      TopEdge     := 0;
      Width       := ImgPtr3^[1].Width;
      Height      := ImgPtr3^[1].Height;
      Depth       := ImgPtr3^[1].Depth;
      ImageData   := ImgPtr3^[1].Data;
      PlanePick   := BYTE(31);
```

```
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;

WITH TabImages[6] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr^[3].Width;
        Height      := ImgPtr^[3].Height;
        Depth       := ImgPtr^[3].Depth;
        ImageData   := ImgPtr^[3].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;

WITH TabImages[7] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr3^[3].Width;
        Height      := ImgPtr3^[3].Height;
        Depth       := ImgPtr3^[3].Depth;
        ImageData   := ImgPtr3^[3].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;
WITH TabImages[8] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr2^[7].Width;
        Height      := ImgPtr2^[7].Height;
        Depth       := ImgPtr2^[7].Depth;
        ImageData   := ImgPtr2^[7].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;
WITH TabImages[9] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr3^[4].Width;
        Height      := ImgPtr3^[4].Height;
        Depth       := ImgPtr3^[4].Depth;
        ImageData   := ImgPtr3^[4].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;
WITH TabImages[10] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr3^[5].Width;
        Height      := ImgPtr3^[5].Height;
        Depth       := ImgPtr3^[5].Depth;
        ImageData   := ImgPtr3^[5].Data;
        PlanePick   := BYTE(31);
        PlaneOnOff  := BYTE(31);
        NextImage := NIL;
END;
WITH TabImages[11] DO
        LeftEdge    := 0;
        TopEdge     := 0;
        Width       := ImgPtr2^[10].Width;
        Height      := ImgPtr2^[10].Height;
        Depth       := ImgPtr2^[10].Depth;
```

```
        ImageData    := ImgPtr2^[10].Data;
        PlanePick    := BYTE(31);
        PlaneOnOff   := BYTE(31);
        NextImage := NIL;
END;

BeginGadgetList();
AddGadgetTextButton (2,4,ADR (" OK "));
Gl := EndGadgetList ();
IF (Gl # NIL) THEN
Win  := CreateWindow (0,0,320,240,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win # NIL) THEN
Win5 := CreateWindow (0,212,320,28,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win5 # NIL) THEN
Win4 := CreateWindow (0,195,275,17,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win4 # NIL) THEN
Win2 := CreateWindow (275,0,45,195,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win2 # NIL) THEN
    Win6 := CreateWindow (285,40,24,102,WIDCMP,WFlags2,NIL,Scr,NIL);
IF (Win6 # NIL) THEN
Win1 := CreateWindow (0,0,275,195,WIDCMP,WFlags,NIL,Scr,NIL);
IF (Win1 # NIL) THEN

Win3 := CreateWindow (275,195,45,17,WIDCMP,WFlags2,Gl,Scr,NIL);
IF (Win3 # NIL) THEN

i := 0;
X := 5;
WHILE (i <= 1) DO
  I := 1;
  Y := 75;
  WHILE (I <= TabNombreInit[i]) AND (I <= 4) DO
    DrawImage (Win1^.RPort^,TabImages[i],X,Y);
    Y := Y - 20;
    I := I + 1;
  END;
  X := X + 40;
  i := i + 1;
END;

i := 2;
X := 5;
WHILE (i <= 3) DO
  I := 1;
  Y := 170;
  WHILE (I <= TabNombreInit[i]) AND (I <= 4) DO
    DrawImage (Win1^.RPort^,TabImages[i],X,Y);
    Y := Y - 20;
    I := I + 1;
  END;
  X := X + 40;
  i := i + 1;
END;

I := 1;
X := 90;
Y := 170;
WHILE (I <= TabNombreInit[4]) AND (I <= 9) DO
    DrawImage (Win1^.RPort^,TabImages[4],X,Y);
    Y := Y - 20;
    I := I + 1;
  END;

I := 1;
X := 120;
Y := 168;
```

```
  WHILE (I <= TabNombreInit[5]) AND (I <= 7) DO
      DrawImage (Win1^.RPort^,TabImages[5],X,Y);
      Y := Y - 25;
      I := I + 1;
    END;

  i := 6;
  X := 150;
  WHILE (i <= 11) DO
    I := 1;
    Y := 170;
    WHILE (I <= TabNombreInit[i]) AND (I <= 9) DO
      DrawImage (Win1^.RPort^,TabImages[i],X,Y);
      Y := Y - 20;
      I := I + 1;
    END;
    X := X + 21;
    i := i + 1;
  END;

  ConvRealToString (Sinit,SommeString,2,Decimal);
  IF (TableauParametres[4] = 3) OR (TableauParametres[4] = 4)
                                OR (TableauParametres[4] = 6)
  THEN
    IF CreateConsole (Win4^)
      THEN
       CopyString (Phrase,"Il vous reste : ");
       ConcatString (Phrase,SommeString);
       ConcatString (Phrase," Francs");
       wSetCursor (Win4^,FALSE);
       wMove (Win4^,1,2);
       PutStr (Win4^,ADR(Phrase));
       DeleteConsole (Win4^);
    END;
END;
    IF (TableauParametres[4] = 4) OR (TableauParametres[4] = 5) OR
       (TableauParametres[4] = 6) THEN
      CopyString (PhrasePron,"il voo rest ");
      ConvPrononcable (SommeString,SomPron);
      ConcatString (PhrasePron,SomPron);
      Res := SayAndReturn (ADR(PhrasePron));

    END;
    CloseSpeech();

  ValeurPrec := 0.;
  Done := FALSE;
  WHILE (NOT Done) DO
    Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
    LOOP
      Msg := GetMsg(Win3^.UserPort^);
      IF (Msg = NIL) THEN EXIT; END;
      ProcIMsg (Wp, Msg);
    END;
  END;
  CloseWindow(Win3^);
  END;

  CloseWindow(Win1^);
  END;
  CloseWindow (Win6^);
  END;
  CloseWindow(Win2^);
  END;
  CloseWindow(Win4^);
  END;
```

```
      CloseWindow(Win5^);
      END;
      CloseWindow(Win^);
      END;
      FreeGadgetList (Gl^);
      END;
      CloseScreen(Scr^);
      END;
END SommeRestanteBillets;


(*  VAR u: INTEGER;

BEGIN
   u := 1;
   SommeRestanteBillets (u);    *)

END SRBillets.
```

Le programme de paramétrisation

```
MODULE TEParam;

FROM SYSTEM IMPORT BYTE, ADR;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOS IMPORT IoErr,CreateDir,FileLock;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenOutputFile,
     CloseOutput;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation, ConcatString,
     StringLength, ConvStringToUpperCase;
FROM VariablesGlobales IMPORT Str3, Str20, Str40, TableauParametres;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM Rasters IMPORT SetRast;
FROM RealInOut IMPORT WriteReal;
FROM RealConversions IMPORT ConvRealToString,RealToStringFormat,
     ConvStringToReal ;
FROM MathLib0 IMPORT real;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Borderless};
  Blanc15 = "               ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
            END;

VAR
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Win,Won,SubWin,Win1,Win2,Win3 : WindowPtr;
  TabImages: ARRAY [0..6] OF Image;
  ImgCount,ImgCount2,Indice,I, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  G1,G1OK,G12,G13 : GadgetPtr;
  Wp,Wp2,WpOK,Wp3 : WindowProc;
  Sig : SignalSet;
  ReqOK : Requester;
  Msg,MsgOK : IntuiMessagePtr;
  Done1,Done2,Efface,Done,DoneOK,OK,Conf,UneSemaine,OKNombre,FenWin,
```

```
        PremPassage : BOOLEAN;
  Int : Gadget;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005 : INTEGER;
  Confirmation : Str40;
  Nom2,Max,L2E1,L5E1,L9E1,L12E1,L15E1,L19E1,L20E1,L21E1,L25E1,L2E2,L7E2,L12E2,
         L17E2,L20E2,L18E2,NomInt,MaxInt : Str20;
  MAX : REAL;
  FileName : Str40;
  Fl : FileLock;
  le : CARDINAL;
  st : Str20;
  For : RealToStringFormat;
  len : INTEGER;
  lene : REAL;

PROCEDURE EnleveBlancs (VAR S : Str20; VAR l : CARDINAL);
VAR Indice1,Indice2 : CARDINAL;
    S2 : Str20;
BEGIN
    Indice1 :=0;
    Indice2 :=0;

    WHILE (Indice1 <= ( StringLength (S) - 1)) DO
      IF (S[Indice1] # " ") THEN
        IF (S[Indice1] ="é") OR (S[Indice1] ="è")
        THEN
            S2[Indice2] := "E";
            Indice2 := Indice2 + 1;
        ELSE
            IF (S[Indice1] # "-") AND (S[Indice1] # "_")
            THEN
            S2[Indice2] := S[Indice1];
            Indice2 := Indice2 + 1;
            END;
        END;
      END;
      Indice1 := Indice1 + 1;
    END;
    S2[Indice2] := S[Indice1];
    l := StringLength (S2);
    CopyString (S,S2);
END EnleveBlancs;

PROCEDURE Reel (VAR St : ARRAY OF CHAR ;  VAR OK : BOOLEAN);
  VAR Indice : CARDINAL;
      Premier : BOOLEAN;
  BEGIN
    Indice := 0;
    OK := TRUE;
    Premier := TRUE;
    WHILE ((Indice < StringLength (St)) AND OK) DO
      IF ((St[Indice] < "0") OR (St[Indice] > "9"))
                THEN IF (((St[Indice] = ",") OR (St[Indice] = ".")) AND Premie
)
                THEN St[Indice] := ".";
                     Indice := Indice + 1;
                     Premier := FALSE;
                ELSE OK := FALSE END;
         ELSE Indice := Indice + 1 END;
    END;
    IF (PremPassage = TRUE) AND (StringLength (St) = 0) THEN OK := FALSE END;
  END Reel;

PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
```

```
      0 : OK := TRUE;DoneOK := TRUE |
      1 : OK := FALSE;DoneOK := TRUE |
      ELSE
  END;
END GadgetHandlerOK;
PROCEDURE OKRequester (FenWin : BOOLEAN);
BEGIN

            BeginGadgetList ();
            GadgetTypeReq := TRUE;
            GlobalGadgetOpt (GadgetFlagsSet {},GadgetActivationSet {EndGadget,
                             RelVerify});
            AddGadgetTextButton (10,40, ADR("OK"));
            AddGadgetTextButton (195,40, ADR("ANNULER"));
            AddGadgetTextButton (15,10, ADR("Introduire la disquette"));
            AddGadgetTextButton (15,21,ADR (" Param"));
            GadgetOpt (GadgetFlagsSet {}, GadgetActivationSet {RelVerify},
                       GadgetMutualExcludeSet {});
            G1OK := EndGadgetList ();
            InitRequester (ReqOK);
            WITH ReqOK DO
                  OlderRequest := NIL;
                  LeftEdge := 185;
                  TopEdge := 20;
                  Width := 300;
                  Height := 60;
                  ReqGadget := G1OK;
                  ReqBorder := NIL;
                  ReqText := NIL;
                  Flags := RequesterFlagsSet {};
                  BackFill := BYTE(10);
                  ReqLayer := NIL;
                  ImageBMap := NIL;
            END;
            IF FenWin
              THEN IF Request (ReqOK, Win^) THEN
                   DoneOK := FALSE;
                   WHILE NOT DoneOK DO
                       Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)})
                       LOOP
                        MsgOK := GetMsg (Win^.UserPort^);
                        IF (MsgOK = NIL) THEN EXIT; END;
                        ProcIMsg (WpOK, MsgOK);
                       END;
                   END;
                 END;
              ELSE IF Request (ReqOK, Win1^) THEN
                   DoneOK := FALSE;
                   WHILE NOT DoneOK DO
                       Sig := Wait(SignalSet{CARDINAL(Win1^.UserPort^.mpSigBit)]
;
                       LOOP
                        MsgOK := GetMsg (Win1^.UserPort^);
                        IF (MsgOK = NIL) THEN EXIT; END;
                        ProcIMsg (WpOK, MsgOK);
                       END;
                   END;
                 END;
             END;
             FreeGadgetList (G1OK^);
END OKRequester;

PROCEDURE GadgetHandler2 (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
```

```
0 : TableauParametres[6] := 0;
    wMove (Win3^,1,2);
    wClrEndLine (Win3^);
    wMove (Win3^,1,2);
    PutStr (Win3^,ADR ("OUI"));
    L2E2 := "OUI" ¦
1 : TableauParametres[6] := 1;
    wMove (Win3^,1,2);
    wClrEndLine (Win3^);
    wMove (Win3^,1,2);
    PutStr (Win3^,ADR ("NON"));
    L2E2 := "NON" ¦
2 : TableauParametres[7] := 0;
    wMove (Win3^,1,7);
    wClrEndLine (Win3^);
    wMove (Win3^,1,7);
    PutStr (Win3^,ADR ("OUI"));
    L7E2 := "OUI" ¦
3 : TableauParametres[7] := 1;
    wMove (Win3^,1,7);
    wClrEndLine (Win3^);
    wMove (Win3^,1,7);
    PutStr (Win3^,ADR ("NON"));
    L7E2 := "NON" ¦
4 : TableauParametres[8] := 0;
    wMove (Win3^,1,12);
    wClrEndLine (Win3^);
    wMove (Win3^,1,12);
    PutStr (Win3^,ADR ("OUI"));
    L12E2 := "OUI" ¦
5 : TableauParametres[8] := 1;
    wMove (Win3^,1,12);
    wClrEndLine (Win3^);
    wMove (Win3^,1,12);
    PutStr (Win3^,ADR ("NON"));
    L12E2 := "NON" ¦
6 : TableauParametres[9] := 0;
    wMove (Win3^,1,17);
    wClrEndLine (Win3^);
    wMove (Win3^,1,17);
    PutStr (Win3^,ADR ("UN JOUR"));
    L17E2 := "UN JOUR";
    L20E2 := "";
    wMove (Win3^,1,20);
    wClrEndLine (Win3^); ¦
7 : TableauParametres[9] := 1;
    wMove (Win3^,1,17);
    wClrEndLine (Win3^);
    wMove (Win3^,1,17);
    PutStr (Win3^,ADR ("UNE SEMAINE"));
    L17E2 := "UNE SEMAINE" ¦
8:  TableauParametres[9] := 2;
    wMove (Win3^,1,17);
    wClrEndLine (Win3^);
    wMove (Win3^,1,17);
    PutStr (Win3^,ADR ("UN MOIS"));
    L17E2 := "UN MOIS";
    L20E2 :="";
    wMove (Win3^,1,20);
    wClrEndLine (Win3^); ¦
9:  IF (CompareString(L17E2,"UNE SEMAINE") = equal) THEN
     TableauParametres[10] := 0;
     wMove (Win3^,1,20);
     wClrEndLine (Win3^);
     wMove (Win3^,1,20);
     PutStr (Win3^,ADR ("LUNDI"));
```

```
          L2OE2 := "LUNDI";
         END |
    10:   IF (CompareString(L17E2,"UNE SEMAINE") = equal) THEN
          TableauParametres[10] := 1;
          wMove (Win3^,1,20);
          wClrEndLine (Win3^);
          wMove (Win3^,1,20);
          PutStr (Win3^,ADR ("MARDI"));
          L2OE2 := "MARDI";
         END |
    11:   IF (CompareString(L17E2,"UNE SEMAINE") = equal) THEN
           TableauParametres[10] := 2;
           wMove (Win3^,1,20);
           wClrEndLine (Win3^);
           wMove (Win3^,1,20);
           PutStr (Win3^,ADR ("MERCREDI"));
           L2OE2 := "MERCREDI";
          END |
    12:   IF (CompareString(L17E2,"UNE SEMAINE") = equal) THEN
            TableauParametres[10] := 3;
            wMove (Win3^,1,20);
            wClrEndLine (Win3^);
            wMove (Win3^,1,20);
            PutStr (Win3^,ADR ("JEUDI"));
            L2OE2 := "JEUDI";
          END |
    13:   IF (CompareString(L17E2,"UNE SEMAINE") = equal) THEN
           TableauParametres[10] := 4;
           wMove (Win3^,1,20);
           wClrEndLine (Win3^);
           wMove (Win3^,1,20);
           PutStr (Win3^,ADR ("VENDREDI"));
           L2OE2 := "VENDREDI";
          END |
    14:   IF (CompareString(L17E2,"UNE SEMAINE") = equal) THEN
           TableauParametres[10] := 5;
           wMove (Win3^,1,20);
           wClrEndLine (Win3^);
           wMove (Win3^,1,20);
           PutStr (Win3^,ADR ("SAMEDI"));
           L2OE2 := "SAMEDI";
          END |
    15:   IF (CompareString(L17E2,"UNE SEMAINE") = equal) THEN
           TableauParametres[10] := 6;
           wMove (Win3^,1,20);
           wClrEndLine (Win3^);
           wMove (Win3^,1,20);
           PutStr (Win3^,ADR ("DIMANCHE"));
           L2OE2 := "DIMANCHE";
          END |
    16:   Done2 := TRUE; Done1 := FALSE |
    17:         FenWin := TRUE;
                OKRequester (FenWin);
                IF OK THEN
                Done := TRUE; Done1 := TRUE; Done2 := TRUE;
                END;
  END;
END GadgetHandler2;



PROCEDURE GadgetHandler3 (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR P : StringInfoPtr;
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
BEGIN
  CASE Gad.GadgetID OF
```

```
        0 : Done := TRUE ;
        1 : Done := TRUE; Efface := FALSE; Done1 := FALSE ;
        2 : FenWin := FALSE;
            OKRequester (FenWin);
            IF OK THEN
               Done := TRUE; Done1 := TRUE; Done2 := TRUE; Efface := FALSE;
            END;
  END;
END GadgetHandler3;


PROCEDURE GadgetHandler (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
VAR P : StringInfoPtr;
VAR PS: POINTER TO ARRAY [0..19] OF CHAR;
BEGIN
  CASE Gad.GadgetID OF
        0 :TableauParametres[0] := 1;
           wMove (Win3^,1,2);
           wClrEndLine (Win3^);
           wMove (Win3^,1,2);
           PutStr (Win3^,ADR ("OUI"));
           L2E1 := "OUI" ;
        1 :TableauParametres[0] := 0;
           wMove (Win3^,1,2);
           wClrEndLine (Win3^);
           wMove (Win3^,1,2);
           PutStr (Win3^,ADR ("NON"));
           L2E1 := "NON" ;
        2 :TableauParametres[1] := 0;
           wMove (Win3^,1,5);
           wClrEndLine (Win3^);
           wMove (Win3^,1,5);
           PutStr (Win3^,ADR ("ECRIT"));
           L5E1 := "ECRIT" ;
        3 :TableauParametres[1] := 2;
           wMove (Win3^,1,5);
           wClrEndLine (Win3^);
           wMove (Win3^,1,5);
           PutStr (Win3^,ADR ("ECRIT + PAROLE"));
           L5E1 := "ECRIT + PAROLE" ;
        4 :TableauParametres[2] := 0;
           wMove (Win3^,1,9);
           wClrEndLine (Win3^);
           wMove (Win3^,1,9);
           PutStr (Win3^,ADR ("CLAVIER"));
           L9E1 := "CLAVIER" ;
        5 :TableauParametres[2] := 1;
           wMove (Win3^,1,9);
           wClrEndLine (Win3^);
           wMove (Win3^,1,9);
           PutStr (Win3^,ADR ("SOURIS"));
           L9E1 := "SOURIS" ;
        6 :TableauParametres[3] := 0;
           wMove (Win3^,1,12);
           wClrEndLine (Win3^);
           wMove (Win3^,1,12);
           PutStr (Win3^,ADR ("CLAVIER"));
           L12E1 := "CLAVIER" ;
        7 :TableauParametres[3] := 1;
           wMove (Win3^,1,12);
           wClrEndLine (Win3^);
           wMove (Win3^,1,12);
           PutStr (Win3^,ADR ("BILLETS"));
           L12E1 := "BILLETS" ;
        8 :TableauParametres[3] := 2;
           wMove (Win3^,1,12);
```

```
    wClrEndLine (Win3^);
    wMove (Win3^,1,12);
    PutStr (Win3^,ADR ("CALCULETTE"));
    L12E1 := "CALCULETTE" ¦
 9 :TableauParametres[12] := 0;
    wMove (Win3^,1,15);
    wClrEndLine (Win3^);
    wMove (Win3^,1,15);
    PutStr (Win3^,ADR ("CLAVIER"));
    L15E1 := "CLAVIER" ¦
 10 :TableauParametres[12] := 1;
    wMove (Win3^,1,15);
    wClrEndLine (Win3^);
    wMove (Win3^,1,15);
    PutStr (Win3^,ADR ("BILLETS"));
    L15E1 := "BILLETS" ¦
 11 :TableauParametres[12] := 2;
    wMove (Win3^,1,15);
    wClrEndLine (Win3^);
    wMove (Win3^,1,15);
    PutStr (Win3^,ADR ("CALCULETTE"));
    L15E1 := "CALCULETTE" ¦
12 :TableauParametres[4] := 0;
    wMove (Win3^,1,19);
    wClrEndLine (Win3^);
    wMove (Win3^,1,20);
    wClrEndLine (Win3^);
    wMove (Win3^,1,21);
    wClrEndLine (Win3^);
    wMove (Win3^,1,19);
    PutStr (Win3^,ADR ("NOMBRES"));
    L19E1 := "NOMBRES";
    L20E1 := Blanc15;
    L20E1 := Blanc15 ¦
 13 :TableauParametres[4] := 1;
    wMove (Win3^,1,19);
    wClrEndLine (Win3^);
    wMove (Win3^,1,20);
    wClrEndLine (Win3^);
    wMove (Win3^,1,21);
    wClrEndLine (Win3^);
    wMove (Win3^,1,19);
    PutStr (Win3^,ADR ("DESSINS"));
    L19E1 := "DESSINS";
    L20E1 := Blanc15;
    L21E1 := Blanc15 ¦
 14:TableauParametres[4] := 3;
    wMove (Win3^,1,19);
    wClrEndLine (Win3^);
    wMove (Win3^,1,20);
    wClrEndLine (Win3^);
    wMove (Win3^,1,21);
    wClrEndLine (Win3^);
    wMove (Win3^,1,19);
    PutStr (Win3^,ADR ("NOMBRES"));
    wMove (Win3^,1,20);
    PutStr (Win3^,ADR ("   +   "));
    wMove (Win3^,1,21);
    PutStr (Win3^,ADR ("DESSINS"));
    L19E1 := "NOMBRES";
    L20E1 := "   +   ";
    L21E1 := "DESSINS"      ¦
 15:TableauParametres[4] := 4;
    wMove (Win3^,1,19);
    wClrEndLine (Win3^);
    wMove (Win3^,1,20);
```

```
            wClrEndLine (Win3^);
            wMove (Win3^,1,21);
            wClrEndLine (Win3^);
            wMove (Win3^,1,19);
            PutStr (Win3^,ADR ("NOMBRES"));
            wMove (Win3^,1,20);
            PutStr (Win3^,ADR ("    +    "));
            wMove (Win3^,1,21);
            PutStr (Win3^,ADR ("PAROLE "));
            L19E1 := "NOMBRES";
            L20E1 := "    +    ";
            L21E1 := "PAROLE "          |
        16:TableauParametres[4] := 5;
            wMove (Win3^,1,19);
            wClrEndLine (Win3^);
            wMove (Win3^,1,20);
            wClrEndLine (Win3^);
            wMove (Win3^,1,21);
            wClrEndLine (Win3^);
            wMove (Win3^,1,19);
            PutStr (Win3^,ADR ("DESSINS"));
            wMove (Win3^,1,20);
            PutStr (Win3^,ADR ("    +    "));
            wMove (Win3^,1,21);
            PutStr (Win3^,ADR ("PAROLE "));
            L19E1 := "DESSINS";
            L20E1 := "    +    ";
            L21E1 := "PAROLE "          |
        17:TableauParametres[4] := 6;
            wMove (Win3^,1,19);
            wClrEndLine (Win3^);
            wMove (Win3^,1,20);
            wClrEndLine (Win3^);
            wMove (Win3^,1,21);
            wClrEndLine (Win3^);
            wMove (Win3^,1,19);
            PutStr (Win3^,ADR ("DESSINS + "));
            wMove (Win3^,1,20);
            PutStr (Win3^,ADR ("PAROLE  + "));
            wMove (Win3^,1,21);
            PutStr (Win3^,ADR ("NOMBRES    "));
            L19E1 := "DESSINS +";
            L20E1 := "PAROLE  +";
            L21E1 := "NOMBRES  "        |
        18: TableauParametres[11] := 1;
            wMove (Win3^,1,25);
            wClrEndLine (Win3^);
            wMove (Win3^,1,25);
            PutStr (Win3^,ADR ("OUI"));
            L25E1 := "OUI" |
        19: TableauParametres[11] := 0;
            wMove (Win3^,1,25);
            wClrEndLine (Win3^);
            wMove (Win3^,1,25);
            PutStr (Win3^,ADR ("NON"));
            L25E1 := "NON" |
        20: Done1 := TRUE; Done2 := FALSE |
        21: Done1 := TRUE; Done2 := TRUE; Done := FALSE |
        22:     FenWin := TRUE;
                OKRequester (FenWin);
                IF OK THEN
                Done := TRUE; Done1 := TRUE; Done2 := TRUE;
                END;
          END;
END GadgetHandler;
```

```
BEGIN

    WITH Wp DO
      procGadgetUp := GadgetHandler;
    END;
    WITH WpOK DO
      procGadgetUp := GadgetHandlerOK;
    END;
    WITH Wp2 DO
      procGadgetUp := GadgetHandler2;
    END;
    WITH Wp3 DO
      procGadgetUp := GadgetHandler3;
    END;
    Scr := CreateScreen (640,240,4,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 0E00H;
      cmap[03] := 0A00H;
      cmap[04] := 0D80H;
      cmap[05] := 0FE0H;
      cmap[06] := 08F0H;
      cmap[07] := 0080H;
      cmap[08] := 00B6H;
      cmap[09] := 00DDH;
      cmap[10] := 00AFH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),16);
      Nom2 := "";
      Max := "";
      L2E1  := "NON";
      L5E1  := "ECRIT + PAROLE";
      L9E1  := "CLAVIER";
      L12E1 := "BILLETS";
      L15E1 := "CALCULETTE";
      L19E1 := "DESSINS +";
      L20E1 := "PAROLE  +";
      L21E1 := "NOMBRES";
      L25E1 := "OUI";
      L2E2  := "OUI";
      L7E2  := "OUI";
      L12E2  := "OUI";
      L17E2 := "UNE SEMAINE";
      L20E2 := "LUNDI";
```

```
      TableauParametres [0] := 0;
      TableauParametres [1] := 2;
      TableauParametres [2] := 0;
      TableauParametres [3] := 1;
      TableauParametres [4] := 6;
      TableauParametres [5] := 0;
      TableauParametres [6] := 0;
      TableauParametres [7] := 0;
      TableauParametres [8] := 0;
      TableauParametres [9] := 1;
      TableauParametres [10] := 0;
      TableauParametres [11] := 1;
      TableauParametres [12] := 2;
      Conf := FALSE;
      PremPassage := TRUE;
      Done := FALSE;
      Done1 := FALSE;
      Done2 := FALSE;
      WHILE (NOT Done) OR (NOT Done1) OR (NOT Done2) DO
       IF NOT Done THEN
        Efface := TRUE;
        WHILE Efface DO
         BeginGadgetList();
         AddGadgetTextButton (10,67,ADR("EFFACER TOUT"));
         AddGadgetTextButton (160,67,ADR("ECRAN SUIVANT"));
         AddGadgetTextButton (599,67,ADR("FIN"));
         G13 := EndGadgetList();
         Win := CreateWindow (0,0,640,145,WIDCMP,WFlags,NIL,Scr,NIL);
         IF (Win # NIL) THEN
             IF CreateConsole (Win^) THEN
                wClrScr (Win^);
                PutStr (Win^,ADR ("Bonjour, les deux écrans suivants vous"));
                PutStr (Win^,ADR (" demandent de sélectionner des valeurs"));
                wMove (Win^,1,2);
                PutStr (Win^,ADR ("pour 12 paramètres. Ensuite, le programme se
));
                PutStr (Win^,ADR (" déroulera en fonction" ));
                wMove (Win^,1,3);
                PutStr (Win^,ADR ("de vos choix. Pour"));
                PutStr (Win^,ADR (" sélectionner une valeur, cliquez sur la cas
"));
                PutStr (Win^,ADR (" adéquate."));
                wMove (Win^,1,4);
                PutStr (Win^,ADR ("A droite se trouvent les valeurs"));
                PutStr (Win^,ADR (" par défaut, puis celles réellement choisies
"));
                wMove (Win^,1,5);
                PutStr (Win^,ADR ("Pour garnir les autres paramètres, cliquez")
;
                PutStr (Win^,ADR (" sur 'ECRAN SUIVANT'."));
                wMove (Win^,1,6);
                PutStr (Win^,ADR ("Quand tous les choix sont"));
                PutStr (Win^,ADR (" faits, cliquez sur 'FIN'."));
                wMove (Win^,1,7);
                PutStr (Win^,ADR ("Avant ces choix, nous vous demandons"));
                PutStr (Win^,ADR (" des renseignements généraux sur"));
                wMove (Win^,1,8);
                PutStr (Win^,ADR ("la personne pour qui les paramètres "));
                PutStr (Win^,ADR (" seront valables."));
                wMove (Win^,1,9);
                PutStr (Win^,ADR ("Après cela, cliquez sur 'ECRAN SUIVANT'"));
                PutStr (Win^,ADR (" et choisissez les paramètres."));

                wMove (Win^,1,12);
                PutStr (Win^,ADR ("Donnez le prénom de l'utilisateur SVP   : "
;
```

```
    IF (CompareString (Nom2,"") <> equal) THEN
            PutStr (Win^,ADR("("));
            PutStr (Win^,ADR(Nom2));
            PutStr (Win^,ADR(") "));
    END;
    NomInt := "";
    IF ((CompareString(Nom2,"") = equal)
             AND (CompareString(NomInt,"") = equal))
      THEN
            WHILE (CompareString(NomInt,"") = equal) DO
             GetStr (Win^,ADR (NomInt),SIZE(NomInt));
            END
      ELSIF  (CompareString(Nom2,"") <> equal)
          THEN   GetStr (Win^,ADR (NomInt),SIZE(NomInt));
    END;
    IF (CompareString (NomInt,"") <> equal) THEN
            CopyString (Nom2,NomInt);
    END;
    ConvStringToUpperCase (Nom2);
    EnleveBlancs (Nom2,le);
    len := le;
    lene := real (le);
    ConvRealToString (lene,st,20,For);
    wMove (Win^,1,13);
    wMove (Win^,1;14);
    PutStr (Win^,ADR ("SVP, donnez le montant d'argent"));
    wMove (Win^,1,15);
    PutStr (Win^,ADR ("maximum que l'utilisateur peut avoir : "));
    IF (CompareString (Max,"") <> equal) THEN
            PutStr (Win^,ADR("("));
            PutStr (Win^,ADR(Max));
            PutStr (Win^,ADR(") "));
    END;
    OKNombre := FALSE; MAX := 0.;
    WHILE NOT OKNombre DO
      IF StringLength (Max) = 0
            THEN wMove (Win^,45,15)
            ELSE wMove (Win^,45 + StringLength (Max) + 2,15)
      END;
      wClrEndLine (Win^);
      GetStr (Win^,ADR (MaxInt),SIZE (MaxInt));
      Reel (MaxInt,OKNombre);
      IF OKNombre THEN
            MAX := ConvStringToReal (MaxInt);
            IF (CompareString (MaxInt,"") <> equal) THEN
               IF (MAX = 0.) THEN
                  OKNombre := FALSE;
               ELSE
                  CopyString (Max,MaxInt);
               END;
            END;
      END;
    END;
    PremPassage := FALSE;
    wSetCursor (Win^,FALSE);


  Win1 := CreateWindow (0,145,640,85,WIDCMP,WFlags2,G13,Scr,NIL);
  IF Win1 # NIL THEN
   Done := FALSE;
   WHILE (NOT Done) DO
       Sig := Wait(SignalSet{CARDINAL(Win1^.UserPort^.mpSigBit)});
       LOOP
         Msg := GetMsg(Win1^.UserPort^);
         IF (Msg = NIL) THEN EXIT; END;
         ProcIMsg (Wp3, Msg);
```

```
            END;
          END;
          CloseWindow (Win1^);
        END;
        DeleteConsole (Win^);
      END;
    CloseWindow (Win^);
    END;
  END; (* du WHILE Efface *)
END;  (* du IF NOT Done  *)


  BeginGadgetList();
  AddGadgetTextButton (200,5,ADR("OUI"));
  AddGadgetTextButton (300,5,ADR("NON"));
  AddGadgetTextButton (200,32,ADR("ECRIT"));
  AddGadgetTextButton (280,32,ADR("ECRIT + PAROLE"));
  AddGadgetTextButton (200,60,ADR("CLAVIER"));
  AddGadgetTextButton (300,60,ADR("SOURIS"));
  AddGadgetTextButton (200,90,ADR("CLAVIER"));
  AddGadgetTextButton (280,90,ADR("BILLETS"));
  AddGadgetTextButton (360,90,ADR("CALCULETTE"));
  AddGadgetTextButton (200,112,ADR("CLAVIER"));
  AddGadgetTextButton (280,112,ADR("BILLETS"));
  AddGadgetTextButton (360,112,ADR("CALCULETTE"));
  AddGadgetTextButton (200,135,ADR("NOMBRES"));
  AddGadgetTextButton (280,135,ADR("DESSINS"));
  AddGadgetTextButton (360,135,ADR("NOMBRES + DESSINS"));
  AddGadgetTextButton (200,150,ADR("NOMBRES + PAROLE"));
  AddGadgetTextButton (360,150,ADR("DESSINS + PAROLE"));
  AddGadgetTextButton (200,165,ADR("DESSINS + PAROLE + NOMBRES"));
  AddGadgetTextButton (200,195,ADR("OUI"));
  AddGadgetTextButton (280,195,ADR("NON"));
  AddGadgetTextButton (160,225,ADR("ECRAN SUIVANT"));
  AddGadgetTextButton (10,225,ADR("ECRAN PRECEDENT"));
  AddGadgetTextButton (599,225,ADR("FIN"));
  G1 := EndGadgetList();
  BeginGadgetList();
  AddGadgetTextButton (200,10,ADR("OUI"));
  AddGadgetTextButton (280,10,ADR("NON"));
  AddGadgetTextButton (200,50,ADR("OUI"));
  AddGadgetTextButton (280,50,ADR("NON"));
  AddGadgetTextButton (200,90,ADR("OUI"));
  AddGadgetTextButton (280,90,ADR("NON"));
  AddGadgetTextButton (200,130,ADR("UN JOUR"));
  AddGadgetTextButton (280,130,ADR("UNE SEMAINE"));
  AddGadgetTextButton (390,130,ADR("UN MOIS"));
  AddGadgetTextButton (200,155,ADR("LUNDI"));
  AddGadgetTextButton (280,155,ADR("MARDI"));
  AddGadgetTextButton (360,155,ADR("MERCREDI"));
  AddGadgetTextButton (440,155,ADR("JEUDI"));
  AddGadgetTextButton (200,170,ADR("VENDREDI"));
  AddGadgetTextButton (280,170,ADR("SAMEDI"));
  AddGadgetTextButton (360,170,ADR("DIMANCHE"));
  AddGadgetTextButton (10,215,ADR("ECRAN PRECEDENT"));
  AddGadgetTextButton (599,215,ADR("FIN"));
  G12 := EndGadgetList();
  IF (G1 # NIL) AND (G12 # NIL) THEN
    IF NOT Done1 THEN
     Win := CreateWindow (0,0,640,240,WIDCMP,WFlags,G1,Scr,NIL);
     Win2 := CreateWindow (0,0,180,220,WIDCMP,WFlags,NIL,Scr,NIL);
     Win3 := CreateWindow (520,0,120,220,WIDCMP,WFlags,NIL,Scr,NIL);
     IF (Win # NIL) AND  (Win2 # NIL)
                    AND  (Win3 # NIL)  THEN
     IF CreateConsole (Win2^) THEN
         PutStr (Win2^,ADR("Voulez-vous un résumé"));
```

```
    wMove (Win2^,1,2);
    PutStr (Win2^,ADR ("sur imprimante ?"));
    wMove (Win2^,1,4);
    PutStr (Win2^,ADR("Les questions sont "));
    wMove (Win2^,1,5);
    PutStr (Win2^,ADR ("posées"));
    PutStr (Win2^,ADR(" par écrit ou "));
    wMove (Win2^,1,6);
    PutStr (Win2^,ADR ("par écrit et parole ?"));
    wMove (Win2^,1,8);
    PutStr (Win2^,ADR ("L'utilisateur entrera"));
    wMove (Win2^,1,9);
    PutStr (Win2^,ADR ("son nom par :"));
    wMove (Win2^,1,11);
    PutStr (Win2^,ADR ("L'utilisateur entrera"));
    wMove (Win2^,1,12);
    PutStr (Win2^,ADR ("les recettes par :"));
    wMove (Win2^,1,14);
    PutStr (Win2^,ADR ("L'utilisateur entrera"));
    wMove (Win2^,1,15);
    PutStr (Win2^,ADR ("les dépenses par :"));
    wMove (Win2^,1,17);
    PutStr (Win2^,ADR ("L'ordinateur montrera"));
    wMove (Win2^,1,18);
    PutStr (Win2^,ADR ("le contenu du porte-"));
    wMove (Win2^,1,19);
    PutStr (Win2^,ADR ("monnaie par nombres,"));
    wMove (Win2^,1,20);
    PutStr (Win2^,ADR ("par dessins, ou par"));
    wMove (Win2^,1,21);
    PutStr (Win2^,ADR ("combinaisons des 2,"));
    wMove (Win2^,1,22);
    PutStr (Win2^,ADR ("avec ou sans parole?"));
    wMove (Win2^,1,24);
    PutStr (Win2^,ADR("Il apparaîtra des"));
    wMove (Win2^,1,25);
    PutStr (Win2^,ADR ("réglettes de couleur"));
    wMove (Win2^,1,26);
    PutStr (Win2^,ADR("représentant les"));
    wMove (Win2^,1,27);
    PutStr (Win2^,ADR ("montants d'argent ?"));
    wSetCursor (Win2^,FALSE);
    DeleteConsole (Win2^);
END;
IF CreateConsole (Win3^) THEN
    wMove (Win3^,1,2);
    PutStr (Win3^,ADR (L2E1));
    wMove (Win3^,1,5);
    PutStr (Win3^,ADR (L5E1));
    wMove (Win3^,1,9);
    PutStr (Win3^,ADR (L9E1));
    wMove (Win3^,1,12);
    PutStr (Win3^,ADR (L12E1));
    wMove (Win3^,1,15);
    PutStr (Win3^,ADR (L15E1));
    wMove (Win3^,1,19);
    PutStr (Win3^,ADR (L19E1));
    wMove (Win3^,1,20);
    PutStr (Win3^,ADR (L20E1));
    wMove (Win3^,1,21);
    PutStr (Win3^,ADR (L21E1));
    wMove (Win3^,1,25);
    PutStr (Win3^,ADR (L25E1));
    wSetCursor (Win3^,FALSE);
WHILE (NOT Done1) DO
  Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
```

```
      LOOP
        Msg := GetMsg(Win^.UserPort^);
        IF (Msg = NIL) THEN EXIT; END;
        ProcIMsg (Wp, Msg);
      END;
    END;
  DeleteConsole (Win3^);
  END;
  CloseWindow(Win^);CloseWindow(Win2^);
  CloseWindow(Win3^);
END;
END; (* du IF NOT Done1 THEN *)
IF NOT Done2 THEN
   Win := CreateWindow (0,0,640,230,WIDCMP,WFlags,G12,Scr,NIL);
   Win2 := CreateWindow (0,0,180,200,WIDCMP,WFlags,NIL,Scr,NIL);
   Win3 := CreateWindow (520,0,120,200,WIDCMP,WFlags,NIL,Scr,NIL);
   IF (Win # NIL) AND  (Win2 # NIL) AND  (Win3 # NIL)  THEN
     IF CreateConsole (Win2^) THEN

       PutStr (Win2^,ADR("Voulez-vous la ligne"));
       wMove (Win2^,1,2);
       PutStr (Win2^,ADR ("du temps ?"));
       wMove (Win2^,1,6);
       PutStr (Win2^,ADR ("L'utilisateur devra-"));
       wMove (Win2^,1,7);
       PutStr (Win2^,ADR ("t-il dire pour quel"));
       wMove (Win2^,1,8);
       PutStr (Win2^,ADR ("poste il a fait une"));
       wMove (Win2^,1,9);
       PutStr (Win2^,ADR ("dépense ?"));
       wMove (Win2^,1,11);
       PutStr (Win2^,ADR ("Voulez-vous l'écran"));
       wMove (Win2^,1,12);
       PutStr (Win2^,ADR ("qui récapitule les"));
       wMove (Win2^,1,13);
       PutStr (Win2^,ADR ("dépenses et recettes"));
       wMove (Win2^,1,16);
       PutStr (Win2^,ADR ("Le cycle est-il un"));
       wMove (Win2^,1,17);
       PutStr (Win2^,ADR ("jour, une semaine"));
       wMove (Win2^,1,18);
       PutStr (Win2^,ADR ("ou un mois"));
       wMove (Win2^,1,20);
       PutStr (Win2^,ADR ("Si cycle = semaine,"));
       wMove (Win2^,1,21);
       PutStr (Win2^,ADR ("donnez le jour de"));
       wMove (Win2^,1,22);
       PutStr (Win2^,ADR ("début de cycle."));
       wMove (Win2^,1,14);
       wSetCursor (Win2^,FALSE);
       DeleteConsole (Win2^);
    END;
    IF CreateConsole (Win3^) THEN
       wMove (Win3^,1,2);
       PutStr (Win3^,ADR (L2E2));
       wMove (Win3^,1,7);
       PutStr (Win3^,ADR (L7E2));
       wMove (Win3^,1,12);
       PutStr (Win3^,ADR (L12E2));
       wMove (Win3^,1,17);
       PutStr (Win3^,ADR (L17E2));
       wMove (Win3^,1,20);
       PutStr (Win3^,ADR (L20E2));
       wSetCursor (Win3^,FALSE);
       WHILE (NOT Done2) DO
        Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSigBit)});
```

```
                LOOP
                  Msg := GetMsg(Win^.UserPort^);
                  IF (Msg = NIL) THEN EXIT; END;
                  ProcIMsg (Wp2, Msg);
                END;
              END;
            DeleteConsole (Win3^);
            END;
          CloseWindow(Win^);CloseWindow(Win2^);CloseWindow(Win3^);
      END;
 (*    END;*)
  END; (* IF NOT Done2 THEN *)
  FreeGadgetList (Gl^);
  FreeGadgetList (Gl2^);
  END; (* du IF Gl^ ... *)
END ;  (* du WHILE *)
CloseScreen(Scr^);
END;   (* du Screen *)
FileName := "Param:";
ConcatString (FileName,"UTILISATEUR");
ConcatString (FileName,".PARAM");
OpenOutputFile (FileName);

IF Done THEN
    I := 0;
    WHILE (I <= 12) DO
      WriteInt (TableauParametres[I],1);
      I := I + 1;
    END;
    CloseOutput;
END;
FileName := "Param:";
ConcatString (FileName,Nom2);
ConcatString (FileName,".MAX");
IF MAX # 0. THEN
  OpenOutputFile (FileName);
  IF Done THEN
      ConvRealToString (MAX,MaxInt,2,Decimal);
      WriteString (MaxInt);
  CloseOutput;
  END;
END;
END TEParam.
```

Le programme d'aide à l'évaluation : les "implementation modules"

```
MODULE EDUC;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM Conversions IMPORT ConvNumberToString;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOS IMPORT IoErr,CreateDir,FileLock;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,ScreenToFront;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenOutputFile,
    OpenInputFile,CloseOutput,CloseInput,Done,ReadInt;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation, ConcatString,
    StringLength, ConvStringToUpperCase,OverwriteWithSubString;
FROM VariablesGlobales IMPORT Str3,Str9, Str20, Str40, TableauParametres,
                              nomuser,Numseance;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM Drawing IMPORT SetAPen,Move,Draw,WritePixel;
FROM RealInOut IMPORT WriteReal,ReadReal;
FROM RealConversions IMPORT ConvRealToString,RealToStringFormat,
    ConvStringToReal ;
FROM MathLib0 IMPORT entier,real;
FROM temps IMPORT InitTemps,EcranTemps,ImpTemps;
FROM Gtemps IMPORT GraphTemps;
FROM Transitions IMPORT Inittransitions,Ecrantransitions;
FROM Parametres IMPORT Paramet;
FROM Critiques IMPORT Ecrancritiques;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Borderless};
  Blanc15 = "               ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
            END;

VAR
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Win,Won,SubWin,Win1,Win2,Win3,Win4 : WindowPtr;
  TabImages: ARRAY [0..6] OF Image;
```

```
  ImgCount,ImgCount2,Indice,I, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  G1,G1OK,G12,G13,G14 : GadgetPtr;
  Wp,Wp2,WpOK,Wp3,Wp4 : WindowProc;
  Sig : SignalSet;
  ReqOK,Req, Req2,Req3 : Requester;
  Msg,MsgOK,Msg2,Msg3,Msg4 : IntuiMessagePtr;
  Done1,Done2,Done3,Done4,Efface,DoneOK,OK,Conf,UneSemaine,OKNombre,FenWin,
        PremPassage : BOOLEAN;
  Int : Gadget;
  B100, B50, B20, B10, P5, P2, P1, P05, P02, P01, P005 : INTEGER;
  Confirmation : Str40;
  Nom2,Max,L2E1,L5E1,L9E1,L12E1,L15E1,L19E1,L20E1,L21E1,L25E1,L2E2,L7E2,L12E2,
         L17E2,L20E2,L18E2,NomInt,MaxInt : Str20;
  MAX : REAL;
  FileName : Str40;
  Fl : FileLock;
  le : CARDINAL;
  st : Str20;
  For : RealToStringFormat;
  len : INTEGER;
  lene : REAL;
  passe : Str20;
  reconnu : BOOLEAN;
  DV,essai : INTEGER;
  DoneNom : BOOLEAN;
  DernVers : REAL;
  TempsSommeInit,TempsSIBillets,TempsCorrection,TempsSommeRec,TempsMomentRec,
  TempsPosteRec,TempsSommeDep,TempsMomentDep,TempsPosteDep,
  TempsTotal : ARRAY [0..50] OF INTEGER;
  LigneBlanc : ARRAY [0..77] OF CHAR;
  LigneSouligne : ARRAY [0..77] OF CHAR;

  NTSI,NTSR,NTSD,NTCO,NTSB,NTMR,NTMD,NTPR,NTPD,NTT,MaxTSI,MinTSI,MaxTSR,
  MinTSR,MaxTSD,MinTSD,MaxTCO,MinTCO,MaxTSB,MinTSB,MaxTMR,MinTMR,MaxTMD,
  MinTMD,MaxTPR,MinTPR,MaxTPD,MinTPD,MaxTT,MinTT : INTEGER;
  CHOIX : ARRAY [0..0] OF CHAR;
  FIN : BOOLEAN;

PROCEDURE EnleveBlancs (VAR S : Str20);
VAR Indice1,Indice2 : CARDINAL;
    S2 : Str20;
BEGIN
    Indice1 :=0;
    Indice2 :=0;

    WHILE (Indice1 <= ( StringLength (S) - 1)) DO
      IF (S[Indice1] # " ") THEN
        IF (S[Indice1] ="é") OR (S[Indice1] ="è")
        THEN
            S2[Indice2] := "E";
            Indice2 := Indice2 + 1;
        ELSE
            IF (S[Indice1] # "-") AND (S[Indice1] # "_")
            THEN
            S2[Indice2] := S[Indice1];
            Indice2 := Indice2 + 1;
            END;
        END;
      END;
      Indice1 := Indice1 + 1;
    END;
    S2[Indice2] := S[Indice1];
    CopyString (S,S2);
END EnleveBlancs;
```

```
PROCEDURE REE1 (VAR St : ARRAY OF CHAR ;  VAR OK : BOOLEAN);
  VAR Indice : CARDINAL;
      Premier : BOOLEAN;
  BEGIN
    Indice := 0;
    OK := TRUE;
    Premier := TRUE;
    WHILE ((Indice < StringLength (St)) AND OK) DO
      IF ((St[Indice] < "0") OR (St[Indice] > "9"))
                  THEN IF (((St[Indice] = ",") OR (St[Indice] = ".")) AND Premie
)
                  THEN St[Indice] := ".";
                       Indice := Indice + 1;
                       Premier := FALSE;
                  ELSE OK := FALSE END;
          ELSE Indice := Indice + 1 END;
    END;
    IF (PremPassage = TRUE) AND (StringLength (St) = 0) THEN OK := FALSE END;
END REE1;


PROCEDURE GadgetHandlerOK (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);
BEGIN
  CASE Gad.GadgetID OF
     0 : OK := TRUE;DoneOK := TRUE |
     1 : OK := FALSE;DoneOK := TRUE |
     ELSE
  END;
END GadgetHandlerOK;




PROCEDURE GadgetHandlerNO (VAR w: Window; VAR Msg : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF
      0 : |
      1 : Done1 := TRUE; Efface := FALSE;|
      2 : Done1 :=TRUE;
     END;
END GadgetHandlerNO;

PROCEDURE GadgetHandler2 (VAR w: Window; VAR Msg2 : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF
      0 : |
      1 : |
      2 : Done2 :=TRUE; |
     ELSE
    END;
END GadgetHandler2;

PROCEDURE GadgetHandler3 (VAR w: Window; VAR Msg3 : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF
      0 : |
      1 : |
      2 : Done3 :=TRUE;
```

```
    END;
END GadgetHandler3;



BEGIN


   WITH Wp DO
      procGadgetUp := GadgetHandlerNO;
   END;

   WITH Wp2 DO
      procGadgetUp := GadgetHandler2;
   END;

   WITH Wp3 DO
      procGadgetUp := GadgetHandler3;
   END;
   LigneBlanc := "                                                              ";
   ConcatString(LigneBlanc,"                                  ");
   LigneSouligne := "     ------------------------------------------------";
   ConcatString(LigneSouligne,"------------------------  ");
    Scr := CreateScreen (640,240,4,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 000FH;
      cmap[03] := 03F1H;
      cmap[04] := 0FE0H;
      cmap[05] := 0F0CH;
      cmap[06] := 00FFH;
      cmap[07] := 0870H;
      cmap[08] := 0E00H;
      cmap[09] := 0F90H;
      cmap[10] := 098FH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),16);
      Win := CreateWindow (0,0,640,240,WIDCMP,WFlags,NIL,Scr,NIL);
      IF (Win # NIL)
      THEN
        IF CreateConsole (Win^) THEN
           essai := 0;
           reconnu := FALSE;
           WHILE (reconnu = FALSE) AND (essai < 4) DO
```

```
    wClrScr (Win^);
    wMove (Win^,25,2);
    PutStr (Win^,ADR("PROGRAMME DE GESTION DES TRACES"));
    wMove (Win^,25,3);
    PutStr (Win^,ADR("------------------------------"));
    wMove (Win^,7,7);
    PutStr (Win^,ADR("Quel est votre mot de passe : "));
    GetStr (Win^,ADR (passe),SIZE (passe));
    IF (CompareString (passe,"") # equal)
    THEN
      EnleveBlancs (passe);
      ConvStringToUpperCase (passe);
      IF (CompareString (passe,"SECRET") = equal)
      THEN reconnu := TRUE;
      END;
      essai := essai + 1;
    END;
END;
IF (reconnu = TRUE)
THEN
  Efface := TRUE;
  WHILE Efface = TRUE DO
     BeginGadgetList();
     GadgetTypeReq := TRUE;
     GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget
           RelVerify});
     AddGadgetTextButton (60,10,ADR("EST-CE BIEN JUSTE ?"));
     GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
           GadgetMutualExcludeSet {});
     AddGadgetTextButton (90,30,ADR("OUI"));
     AddGadgetTextButton (170,30,ADR("NON"));
     G1 := EndGadgetList();
     InitRequester (Req);
     WITH Req DO
        OlderRequest := NIL;
        LeftEdge := 140;
        TopEdge := 140;
        Width := 300;
        Height := 60;
        ReqGadget := G1;
        ReqBorder := NIL;
        ReqText := NIL;
        Flags := RequesterFlagsSet {};
        BackFill := BYTE(8);
        ReqLayer := NIL;
        ImageBMap := NIL;
      END;

      BeginGadgetList();
      GadgetTypeReq := TRUE;
      GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadge

           RelVerify});
      AddGadgetTextButton (30,10,ADR
         ("INSEREZ LA DISQUETTE PARAM "));
      GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
           GadgetMutualExcludeSet {});
      AddGadgetTextButton (80,25,ADR("DE CET UTILISATEUR"));
      GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
           GadgetMutualExcludeSet {});
      AddGadgetTextButton (120,45,ADR(" OK "));
      G12 := EndGadgetList();
      InitRequester (Req2);
      WITH Req2 DO
        OlderRequest := NIL;
        LeftEdge := 140;
```

```
  TopEdge := 140;
  Width := 300;
  Height := 60;
  ReqGadget := G12;
  ReqBorder := NIL;
  ReqText := NIL;
  Flags := RequesterFlagsSet {};
  BackFill := BYTE(8);
  ReqLayer := NIL;
  ImageBMap := NIL;
END;

BeginGadgetList();
GadgetTypeReq := TRUE;
GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadge

      RelVerify});
AddGadgetTextButton (30,10,ADR
   ("JE NE TROUVE PAS CE NOM "));
GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
      GadgetMutualExcludeSet {});
AddGadgetTextButton (80,25,ADR("ESSAYEZ ENCORE"));
GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
      GadgetMutualExcludeSet {});
AddGadgetTextButton (120,45,ADR(" OK "));
G13 := EndGadgetList();
InitRequester (Req3);
WITH Req3 DO
  OlderRequest := NIL;
  LeftEdge := 140;
  TopEdge := 140;
  Width := 300;
  Height := 60;
  ReqGadget := G13;
  ReqBorder := NIL;
  ReqText := NIL;
  Flags := RequesterFlagsSet {};
  BackFill := BYTE(8);
  ReqLayer := NIL;
  ImageBMap := NIL;
END;

wMove (Win^,7,12);
PutStr(Win^,ADR("Quel est le nom de l'utilisateur : "));
wMove(Win^,42,12);
PutStr(Win^,ADR("                          "));
wMove(Win^,42,12);
wSetCursor (Win^,TRUE);
GetStr (Win^,ADR (nomuser), SIZE (nomuser));
EnleveBlancs (nomuser);
ConvStringToUpperCase (nomuser);
wSetCursor (Win^,FALSE);
IF Request (Req, Win^)
THEN
   Done1 := FALSE;
   WHILE (NOT Done1) DO
      Sig := Wait(SignalSet{CARDINAL
                  (Win^.UserPort^.mpSigBit)});
      LOOP
         Msg := GetMsg(Win^.UserPort^);
         IF (Msg = NIL) THEN EXIT; END;
         ProcIMsg (Wp, Msg);
      END;
   END;
   IF Efface = FALSE (* càd il confirme le nom *)
   THEN
```

```
                    IF Request (Req2, Win^)
                    THEN
                      Done2 := FALSE;
                      WHILE (NOT Done2) DO
                        Sig := Wait(SignalSet{CARDINAL(Win^.UserPort^.mpSig
Bit)});
                        LOOP
                          Msg2 := GetMsg(Win^.UserPort^);
                          IF (Msg2 = NIL) THEN EXIT; END;
                          ProcIMsg (Wp2, Msg2);
                        END;
                      END;
                    ELSE WriteString("requester pas créé");
                    END;
                    FileName := "Param:";
                    ConcatString (FileName,nomuser);
                    ConcatString (FileName,".DV");

                    OpenInputFile (FileName);
                    ScreenToFront(Scr^);
                    IF Done
                    THEN ReadReal (DernVers);
                         IF (DernVers > 49.)
                         THEN DV := 49;
                         ELSE DV := entier (DernVers);
                         END;
                         CloseInput;

                    ELSE
                      Efface := TRUE;
                      IF Request (Req3, Win^)
                      THEN
                        Done3 := FALSE;
                        WHILE (NOT Done3) DO
                         Sig := Wait(SignalSet
                                     {CARDINAL(Win^.UserPort^.mpSigBit)});
                         LOOP
                          Msg3 := GetMsg(Win^.UserPort^);
                          IF (Msg3 = NIL) THEN EXIT; END;
                          ProcIMsg (Wp3, Msg3);
                         END;
                        END;
                      ELSE WriteString("requester pas créé");
                      END;
                    END;

              END; (* if efface *)
            ELSE WriteString ("requester pas créé");
            END;

            FreeGadgetList (Gl^);
            FreeGadgetList (Gl2^);
            FreeGadgetList (Gl3^);
        END; (*While efface*)
        FIN := FALSE;
        WHILE (NOT FIN) DO
          wClrScr(Win^);
          wMove(Win^,25,2);
          PutStr(Win^,ADR("PROGRAMME DE GESTION DES TRACES"));
          wMove(Win^,25,3);
          PutStr(Win^,ADR("------------------------------"));
          wMove(Win^,10,7);
          PutStr(Win^,ADR("1.] Tableau des temps"));
          wMove(Win^,10,9);
          PutStr(Win^,ADR("2.] Graphique des temps"));
          wMove(Win^,10,11);
```

```
PutStr(Win^,ADR("3.] Matrice des transitions"));
wMove(Win^,10,13);
PutStr(Win^,ADR("4.] Transitions critiques"));
wMove(Win^,10,15);
PutStr(Win^,ADR("5.] Traces d'utilisation"));
wMove(Win^,10,17);
PutStr(Win^,ADR("6.] Quitter"));
wMove(Win^,15,20);
wSetCursor(Win^,TRUE);
PutStr(Win^,ADR("ENTRER VOTRE CHOIX [1->6] : "));
GetStr(Win^,ADR(CHOIX),SIZE(CHOIX));
wSetCursor(Win^,FALSE);
IF (CompareString(CHOIX,"") # equal)
THEN
 IF (CompareString(CHOIX,"1") = equal)
 THEN
    wClrScr(Win^);
    wMove(Win^,30,15);
    PutStr(Win^,ADR("CHARGEMENT EN COURS"));
    InitTemps (DV,TempsSommeInit,TempsSommeRec,TempsSommeDep,
               TempsSIBillets,TempsCorrection,TempsMomentRec,
               TempsMomentDep,TempsPosteRec,TempsPosteDep,
               TempsTotal);

    EcranTemps (DV,MaxTSI,MaxTSR,MaxTSD,MaxTSB,MaxTCO,
               MaxTMR,MaxTMD,MaxTPR,MaxTPD,MaxTT,TempsSommeInit,
               TempsSommeRec,TempsSommeDep,TempsSIBillets,
               TempsCorrection,TempsMomentRec,TempsMomentDep,
               TempsPosteRec,TempsPosteDep,TempsTotal);
 END;
 IF (CompareString(CHOIX,"2") = equal)
 THEN
    wClrScr(Win^);
    wMove(Win^,30,15);
    PutStr(Win^,ADR("CHARGEMENT EN COURS"));
    InitTemps (DV,TempsSommeInit,TempsSommeRec,TempsSommeDep,
               TempsSIBillets,TempsCorrection,TempsMomentRec,
               TempsMomentDep,TempsPosteRec,TempsPosteDep,
               TempsTotal);

    GraphTemps (DV,MaxTSI,MaxTSR,MaxTSD,MaxTSB,MaxTCO,MaxTMR,
                MaxTMD,MaxTPR,MaxTPD,MaxTT,TempsSommeInit,
                TempsSommeRec,TempsSommeDep,TempsSIBillets,
                TempsCorrection,TempsMomentRec,TempsMomentDep,
                TempsPosteRec,TempsPosteDep,TempsTotal);
 END;

 IF (CompareString(CHOIX,"3") = equal)
 THEN
    wClrScr(Win^);
    wMove(Win^,30,15);
    PutStr(Win^,ADR("CHARGEMENT EN COURS"));
    Inittransitions(DernVers,FALSE);
    Ecrantransitions;
 END;

 IF (CompareString(CHOIX,"4") = equal)
 THEN
    wClrScr(Win^);
    Inittransitions(DernVers,TRUE);
    Ecrancritiques;
 END;

 IF (CompareString(CHOIX,"5") = equal)
 THEN
    wClrScr(Win^);
```

```
                    wMove(Win^,30,15);
                    PutStr(Win^,ADR("CHARGEMENT EN COURS"));
                    Paramet (DernVers);
                 END;

                 IF (CompareString(CHOIX,"6") = equal)
                 THEN
                    FIN := TRUE;
                 END;

               END;
             END;

          END; (* if reconnu *)

        DeleteConsole (Win^);
        END;(*console*)
        CloseWindow(Win^);
      END;(* if Win*)
      CloseScreen (Scr^);
    END;
END EDUC.
```

```
IMPLEMENTATION MODULE Gtemps;


FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM Conversions IMPORT ConvNumberToString;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOS IMPORT IoErr,CreateDir,FileLock;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,ScreenToFront;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM Text IMPORT Text;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenOutputFile,
     OpenInputFile,CloseOutput,CloseInput,Done,ReadInt;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation, ConcatString,
     StringLength, ConvStringToUpperCase,OverwriteWithSubString;
FROM VariablesGlobales IMPORT Str3,Str9, Str20, Str40, TableauParametres
                              ,nomuser;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM Drawing IMPORT SetAPen,Move,Draw,WritePixel;
FROM RealInOut IMPORT WriteReal,ReadReal;
FROM RealConversions IMPORT ConvRealToString,RealToStringFormat,
     ConvStringToReal ;
FROM MathLib0 IMPORT entier,real;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Borderless};
  Blanc15 = "               ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
            END;

VAR
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Win,Win1,Win2,Win3,Win4 : WindowPtr;
  Indice,I, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  G1,G1OK,G12,G13,G14 : GadgetPtr;
```

```
  Wp,Wp2,WpOK,Wp3,Wp4 : WindowProc;
  Sig : SignalSet;
  ReqOK,Req, Req2,Req3 : Requester;
  Msg,MsgOK,Msg2,Msg3,Msg4 : IntuiMessagePtr;
  Done1,Done2,Done3,Done4,Efface,DoneOK,OK,Conf,UneSemaine,OKNombre,FenWin,
        PremPassage : BOOLEAN;
  Int : Gadget;
  Confirmation : Str40;
  Nom2,Max,NomInt,MaxInt : Str20;
  MAX : REAL;
  FileName : Str40;
  Fl : FileLock;
  le : CARDINAL;
  st : Str20;
  For : RealToStringFormat;
  len : INTEGER;
  lene : REAL;
  passe : Str20;
  reconnu : BOOLEAN;
  DV,essai : INTEGER;
  DoneNom : BOOLEAN;
  DernVers : REAL;
  TempsSommeInit,TempsSIBillets,TempsCorrection,TempsSommeRec,TempsMomentRec,
  TempsPosteRec,TempsSommeDep,TempsMomentDep,TempsPosteDep,
  TempsTotal : ARRAY [0..50] OF INTEGER;
  LigneBlanc : ARRAY [0..77] OF CHAR;
  LigneSouligne : ARRAY [0..77] OF CHAR;

  NTSI,NTSR,NTSD,NTCO,NTSB,NTMR,NTMD,NTPR,NTPD,NTT,MaxTSI,MinTSI,MaxTSR,
  MinTSR,MaxTSD,MinTSD,MaxTCO,MinTCO,MaxTSB,MinTSB,MaxTMR,MinTMR,MaxTMD,
  MinTMD,MaxTPR,MinTPR,MaxTPD,MinTPD,MaxTT,MinTT : INTEGER;


PROCEDURE GadgetHandler4 (VAR W:Window; VAR Msg4 : MsgData; VAR Gad : Gadget);

BEGIN

   Done4 := TRUE;

END GadgetHandler4;

PROCEDURE GraphTemps (VAR DV,MaxTSI,MaxTSR,MaxTSD,MaxTSB,MaxTCO,
                      MaxTMR,MaxTMD,MaxTPR,MaxTPD,MaxTT:INTEGER ;
                      VAR TempsSommeInit,TempsSommeRec,TempsSommeDep,
                      TempsSIBillets,TempsCorrection,TempsMomentRec,
                      TempsMomentDep,TempsPosteRec,TempsPosteDep,
                      TempsTotal : ARRAY  OF INTEGER);



VAR Tircinq, Ecrcinq, Tirun, Ecrun, Blanc : BOOLEAN;
    Nbpixels,Rmax,Maxint,Maxre,Echelle,Nbint,Distint,Ibis,Hautint,Ordint,
     Absre :REAL;
    Maxent,Max,Min,Indse,Nb,Fin,Dist,I,J,Haut,Abs,Ord,MaxT,Abstxt,en : INTEGER;
    Istr : Str20;
    Rp :RastPortPtr;
    Maxent2 : INTEGER;
    Maxint2 : REAL;
    Tirquinze, Ecrquinze : BOOLEAN;

    Se,Longecr : INTEGER;
    MTSI,MTSR,MTSD,MTCO,MTSB,MTMR,MTMD,MTPR,MTPD,MTT,REE,RE1 : REAL;
    Prem : BOOLEAN;
    StrTemps,StrSeance,StrMoy,StrMax,StrMin : Str20;
    Ligne,Ligne1,Ligne2,Ligne3 : ARRAY [0..77] OF CHAR;
```

```
BEGIN

    WITH Wp4 DO
       procGadgetUp := GadgetHandler4;
    END;

   LigneBlanc := "                                                                ";
   ConcatString(LigneBlanc,"                                  ");
   LigneSouligne := "     ------------------------------------------------";
   ConcatString(LigneSouligne,"-------------------------  ");

    Scr := CreateScreen (640,240,4,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 000FH;
      cmap[03] := 03F1H;
      cmap[04] := 0FE0H;
      cmap[05] := 0F0CH;
      cmap[06] := 00FFH;
      cmap[07] := 0870H;
      cmap[08] := 0E00H;
      cmap[09] := 0F90H;
      cmap[10] := 098FH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
     LoadRGB4 (Scr^.ViewPort,ADR (cmap),16);

     Se := 0;
     MaxTSI :=0;
     MaxTSR :=0;
     MaxTSD :=0;
     MaxTCO :=0;
     MaxTSB :=0;
     MaxTMR :=0;
     MaxTMD :=0;
     MaxTPR :=0;
     MaxTPD :=0;
     MaxTT :=0;
     WHILE (Se <= DV) DO
       IF (TempsSommeInit[Se] > MaxTSI)
       THEN MaxTSI := TempsSommeInit[Se];
       END;
       IF (TempsSommeRec[Se] > MaxTSR)
       THEN MaxTSR := TempsSommeRec[Se];
       END;
       IF (TempsSommeDep[Se] > MaxTSD)
```

```
      THEN MaxTSD := TempsSommeDep[Se];
      END;
      IF (TempsCorrection[Se] > MaxTCO)
      THEN MaxTCO := TempsCorrection[Se];
      END;
      IF (TempsSIBillets[Se] > MaxTSB)
      THEN MaxTSB := TempsSIBillets[Se];
      END;
      IF (TempsMomentRec[Se] > MaxTMR)
      THEN MaxTMR := TempsMomentRec[Se];
      END;
      IF (TempsMomentDep[Se] > MaxTMD)
      THEN MaxTMD := TempsMomentDep[Se];
      END;
      IF (TempsPosteRec[Se] > MaxTPR)
      THEN MaxTPR := TempsPosteRec[Se];
      END;
      IF (TempsPosteDep[Se] > MaxTPD)
      THEN MaxTPD := TempsPosteDep[Se];
      END;
      IF (TempsTotal[Se] > MaxTT)
      THEN MaxTT := TempsTotal[Se];
      END;
      Se := Se + 1;
    END;

   Win2 := CreateWindow(0,0,640,240,WIDCMP,WFlags2,NIL,Scr,NIL);
   IF (Win2 # NIL)
   THEN
    IF CreateConsole(Win2^)
    THEN

     wClrScr(Win2^);
     wSetCursor (Win2^,FALSE);
     wMove(Win2^,30,1);
     PutStr(Win2^,ADR("TEMPS D'EXECUTION"));
     wMove(Win2^,30,2);
     PutStr(Win2^,ADR("-----------------"));
     wMove(Win2^,5,4);
     PutStr(Win2^,ADR("Les écrans suivants vous présentent différents temps"))

     wMove(Win2^,3,10);
     PutStr(Win2^,ADR("- TSI = Temps nécessaire pour saisir la somme initiale"
);
     wMove(Win2^,3,11);
     PutStr(Win2^,ADR("- TSR = Temps nécessaire pour saisir le montant d'une r
cette"));
     wMove(Win2^,3,12);
     PutStr(Win2^,ADR("- TSD = Temps nécessaire pour saisir le montant d'une d
pense"));
     wMove(Win2^,3,13);
     PutStr(Win2^,ADR("- TCO = Temps nécessaire pour corriger le montant initi
l"));
     wMove(Win2^,3,14);
     PutStr(Win2^,ADR("- TSB = Temps nécessaire pour cliquer sur un billet"));
     wMove(Win2^,3,15);
     PutStr(Win2^,ADR("        lors de la saisie de la somme initiale"));
     wMove(Win2^,3,16);
     PutStr(Win2^,ADR("- TMR = Temps nécessaire pour saisir le moment d'une re
ette"));
     wMove(Win2^,3,17);
     PutStr(Win2^,ADR("- TMD = Temps nécessaire pour saisir le moment d'une de
ense"));
     wMove(Win2^,3,18);
     PutStr(Win2^,ADR("- TPR = Temps nécessaire pour saisir le poste d'une re
tte"));
```

```
      wMove(Win2^,3,19);
      PutStr(Win2^,ADR("- TPD = Temps nécessaire pour saisir le poste d'une dépe
nse"));
      wMove(Win2^,3,20);
      PutStr(Win2^,ADR("- TT  = Temps total de la séance"));
      DeleteConsole(Win2^);
     END;
     BeginGadgetList ();
     AddGadgetTextButton (1,1,ADR(" OK "));
     G14 := EndGadgetList ();
     Win3 := CreateWindow (312,226,56,12,WIDCMP,WFlags2,G14,Scr,NIL);
     Done4 := FALSE;
     WHILE (NOT Done4) DO
          Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
          LOOP
           Msg4 := GetMsg(Win3^.UserPort^);
           IF (Msg4 = NIL) THEN EXIT; END;
           ProcIMsg (Wp4, Msg4);
          END;
     END;
     CloseWindow(Win3^);

    END;



    Win2 := CreateWindow(0,0,640,240,WIDCMP,WFlags2,NIL,Scr,NIL);
    IF (Win2 # NIL)
    THEN
     IF CreateConsole(Win2^)
     THEN
      Rp := Win2^.RPort;
      wClrScr(Win2^);
      wSetCursor (Win2^,FALSE);
      SetAPen (Rp^,1);
      Nbpixels := 150.;
      Tircinq := FALSE;
      Ecrcinq := FALSE;
      Tirun := FALSE;
      Ecrun := FALSE;
      Rmax := real(MaxTT);
      Maxint := Rmax/60.;
      Maxent := entier(Maxint);
      Maxent := Maxent + 1;
      Max := Maxent * 60;
      Maxre := real(Max);
      Min := 0;
      Echelle := Nbpixels/Maxre;
      IF Maxent > 30
      THEN Tircinq := TRUE
      ELSE Ecrcinq := TRUE;
           IF Maxent > 6
           THEN Tirun := TRUE
           ELSE Ecrun := TRUE
           END;
      END;
      Indse := 0;
      WHILE Indse <= DV DO
        wMove(Win2^,35,1);
        PutStr(Win2^,ADR("Temps total"));
        wMove(Win2^,35,2);
        PutStr(Win2^,ADR("-----------"));
        IF Indse # 0
        THEN Indse := Indse - 1;
```

```
END;
IF (DV - Indse + 1) >21
THEN Nb := 21
ELSE Nb := DV - Indse + 1
END;
Fin := Nb + Indse - 1;
Nbint := real(Nb);
Distint := 525./Nbint;
Dist := entier(Distint);
Move(Rp^,85,35);
Draw(Rp^,85,195);
Draw(Rp^,615,195);
Draw(Rp^,611,192);
Move(Rp^,615,195);
Draw(Rp^,611,198);
Move(Rp^,82,39);
Draw(Rp^,85,35);
Draw(Rp^,88,39);
wMove(Win2^,10,4);
PutStr(Win2^,ADR("Min."));
wMove(Win2^,74,26);
PutStr(Win2^,ADR("Séance"));
wMove(Win2^,8,25);
PutStr(Win2^,ADR("0"));
Move(Rp^,82,195);
Draw(Rp^,85,195);
J := 1;
I := 1;
WHILE I <= Maxent DO
  WHILE (J <= 4) AND (I <= Maxent) DO
    IF (Tirun = TRUE) OR (Ecrun = TRUE)
    THEN
      Ibis := real(I);
      Hautint := Ibis * Echelle * 60.;
      Haut := entier(Hautint);
      Move(Rp^,82,195 - Haut);
      Draw(Rp^,85,195 - Haut);
      IF Ecrun = TRUE
      THEN Hautint := Hautint/8.;
           Hautint := Hautint + 0.5;
           Haut := entier(Hautint);
           wMove(Win2^,8,25 - Haut);
           ConvRealToString(Ibis,Istr,0,Decimal);
           PutStr(Win2^,ADR(Istr));
      END;
    END;
    J := J + 1;
    I := I + 1;
  END;
  IF (I <= Maxent)
  THEN
    IF (Tircinq = TRUE) OR (Ecrcinq = TRUE)
    THEN
      Ibis := real(I);
      Hautint := Ibis * Echelle * 60.;
      Haut := entier(Hautint);
      Move(Rp^,80,195 - Haut);
      Draw(Rp^,85,195 - Haut);
      IF Ecrcinq = TRUE
      THEN Hautint := Hautint/8.;
           Hautint := Hautint + 0.5;
           Haut := entier(Hautint);
           wMove(Win2^,8,25 - Haut);
           ConvRealToString(Ibis,Istr,0,Decimal);
           PutStr(Win2^,ADR(Istr));
      END;
```

```
    END;
    J := J + 1;
    I := I + 1;
  END;
  WHILE (J <= 9) AND (I <= Maxent) DO
    IF (Tirun = TRUE) OR (Ecrun = TRUE)
    THEN
      Ibis := real(I);
      Hautint := Ibis * Echelle * 60.;
      Haut := entier(Hautint);
      Move(Rp^,82,195 - Haut);
      Draw(Rp^,85,195 - Haut);
      IF Ecrun = TRUE
      THEN Hautint := Hautint/8.;
           Hautint := Hautint + 0.5;
           Haut := entier(Hautint);
           wMove(Win2^,8,25 - Haut);
           ConvRealToString(Ibis,Istr,0,Decimal);
           PutStr(Win2^,ADR(Istr));
      END;
    END;
    J := J + 1;
    I := I + 1;
  END;
  IF (I <= Maxent)
  THEN
      Ibis := real(I);
      Hautint := Ibis * Echelle * 60.;
      Haut := entier(Hautint);
      Move(Rp^,80,195 - Haut);
      Draw(Rp^,85,195 - Haut);
      Hautint := Hautint/8.;
      Hautint := Hautint + 0.5;
      Haut := entier(Hautint);
      wMove(Win2^,8,25 - Haut);
      ConvRealToString(Ibis,Istr,0,Decimal);
      PutStr(Win2^,ADR(Istr));
      J := 1;
      I := I + 1;
  END;
END;
I := Indse;
Abs := 85       ;
WHILE (I <= DV) AND (I <= Fin) DO
  Move(Rp^,Abs,198);
  Draw(Rp^,Abs,195);
  Absre := real(Abs);
  Absre := Absre/8.;
  Absre := Absre + 0.5;
  Abstxt := entier(Absre);
  wMove(Win2^,Abstxt,26);
  Ibis := real(I+1);
  ConvRealToString(Ibis,Istr,0,Decimal);
  PutStr(Win2^,ADR(Istr));
  I := I + 1;
  Abs := Abs + Dist;
END;
SetAPen (Rp^,4);
IF DV = 0
THEN Abs := 85        ;
     IF TempsTotal[Indse] # 0
     THEN Ordint := real(TempsTotal[Indse]);
          Ordint := Ordint * Echelle;
          Ord := entier(Ordint);
          Move(Rp^,Abs-2,195 - Ord);
          Draw(Rp^,Abs,195 - Ord);
```

```
        END;
        Indse := Indse + 1;
    ELSE
    Abs := 85 ;
    IF TempsTotal[Indse] # 0
    THEN Ordint := real(TempsTotal[Indse]);
        Ordint := Ordint * Echelle;
        Ord := entier(Ordint);
        Move(Rp^,Abs,195 - Ord);
        en:= WritePixel(Rp^,Abs,(195 - Ord));
        Blanc := FALSE;
    ELSE Blanc := TRUE;
    END;
    Abs := Abs + Dist;
    Indse := Indse + 1;
    WHILE (Indse <= DV) AND (Indse <= Fin) DO
      IF Blanc = FALSE
      THEN IF TempsTotal[Indse] # 0
           THEN Ordint := real(TempsTotal[Indse]);
                Ordint := Ordint * Echelle;
                Ord := entier(Ordint);
                Draw(Rp^,Abs,195 - Ord);
                en := WritePixel(Rp^,Abs,195 - Ord);
           ELSE Blanc := TRUE;
           END;
      ELSE IF TempsTotal[Indse] # 0
           THEN Ordint := real(TempsTotal[Indse]);
                Ordint := Ordint * Echelle;
                Ord := entier(Ordint);
                Move(Rp^,Abs,195 - Ord);
                en := WritePixel(Rp^,Abs,195 - Ord);
                Blanc := FALSE;
            END;
      END;
      Abs := Abs + Dist;
      Indse := Indse + 1;
    END;
    END;

    BeginGadgetList ();
    AddGadgetTextButton (1,1,ADR(" OK "));
    Gl4 := EndGadgetList ();
    Win3 := CreateWindow (312,226,56,12,WIDCMP,WFlags2,Gl4,Scr,NIL);
    Done4 := FALSE;
    WHILE (NOT Done4) DO
      Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
      LOOP
        Msg4 := GetMsg(Win3^.UserPort^);
        IF (Msg4 = NIL) THEN EXIT; END;
        ProcIMsg (Wp4, Msg4);
      END;
    END;
    CloseWindow(Win3^);
    FreeGadgetList (Gl4^);
    wClrScr (Win2^);


END;

Tirquinze := FALSE;
Ecrquinze := FALSE;
Tircinq := FALSE;
Ecrcinq := FALSE;
Tirun := FALSE;
Ecrun := FALSE;
MaxT := 0;
```

```
IF (MaxTSI > MaxT)
THEN MaxT := MaxTSI;
END;
IF (MaxTSB > MaxT)
THEN MaxT := MaxTSB;
END;
IF (MaxTSR > MaxT)
THEN MaxT := MaxTSR;
END;
IF (MaxTSD > MaxT)
THEN MaxT := MaxTSD;
END;
IF (MaxTCO > MaxT)
THEN MaxT := MaxTCO;
END;
IF (MaxTMR > MaxT)
THEN MaxT := MaxTMR;
END;
IF (MaxTMD > MaxT)
THEN MaxT := MaxTMD;
END;
IF (MaxTPR > MaxT)
THEN MaxT := MaxTPR;
END;
IF (MaxTPD > MaxT)
THEN MaxT := MaxTPD;
END;
Rmax := real(MaxT);
Maxint := Rmax/15.;
Maxint2 := Rmax /60.;
Maxent2 := entier(Maxint2);
Maxent := entier(Maxint);
Maxent := Maxent + 1;
Max := Maxent * 15;
Maxre := real(Max);
Min := 0;
Echelle := Nbpixels/Maxre;
IF Maxent > 120
THEN Tircinq := TRUE
ELSE Ecrcinq := TRUE;
   IF Maxent > 24
   THEN Tirun := TRUE
   ELSE Ecrun := TRUE
   END;
END;
IF (Maxent <= 16) AND (Maxent >= 8)
THEN Tirquinze := TRUE;
END;
IF (Maxent < 8)
THEN Ecrquinze := TRUE;
END;
SetAPen (Rp^,1);
wClrScr(Win2^);
Indse := 0;
WHILE (Indse <= DV) DO
  IF Indse # 0
  THEN Indse := Indse - 1;
  END;
  IF (DV - Indse + 1) >21
  THEN Nb := 21
  ELSE Nb := DV - Indse + 1
  END;
Fin := Nb + Indse - 1;
Nbint := real(Nb);
Distint := 525./Nbint;
Dist := entier(Distint);
```

```
Move(Rp^,85,35);
Draw(Rp^,85,195);
Draw(Rp^,615,195);
Draw(Rp^,611,192);
Move(Rp^,615,195);
Draw(Rp^,611,198);
Move(Rp^,82,39);
Draw(Rp^,85,35);
Draw(Rp^,88,39);
SetAPen(Rp^,2);
Move(Rp^,160,8);
Text(Rp^,ADR("TSI"),StringLength("TSI"));
SetAPen(Rp^,3);
Move(Rp^,160,20);
Text(Rp^,ADR("TSR"),StringLength("TSR"));
SetAPen(Rp^,4);
Move(Rp^,160,32);
Text(Rp^,ADR("TSD"),StringLength("TSD"));
SetAPen(Rp^,5);
Move(Rp^,320,8);
Text(Rp^,ADR("TCO"),StringLength("TCO"));
SetAPen(Rp^,6);
Move(Rp^,320,20);
Text(Rp^,ADR("TSB"),StringLength("TSB"));
SetAPen(Rp^,7);
Move(Rp^,320,32);
Text(Rp^,ADR("TMR"),StringLength("TMR"));
SetAPen(Rp^,8);
Move(Rp^,480,8);
Text(Rp^,ADR("TMD"),StringLength("TMD"));
SetAPen(Rp^,9);
Move(Rp^,480,20);
Text(Rp^,ADR("TPR"),StringLength("TPR"));
SetAPen(Rp^,10);
Move(Rp^,480,32);
Text(Rp^,ADR("TPD"),StringLength("TPD"));

SetAPen(Rp^,1);
wMove(Win2^,10,4);
IF (Ecrquinze = TRUE)
THEN
 PutStr(Win2^,ADR("Sec."));
ELSE
 PutStr(Win2^,ADR("Min."));
END;
wMove(Win2^,74,26);
PutStr(Win2^,ADR("Séance"));
wMove(Win2^,8,25);
PutStr(Win2^,ADR("0"));
Move(Rp^,82,195);
Draw(Rp^,85,195);
IF (Ecrquinze = FALSE)
THEN
J := 1;
I := 1;
WHILE I <= Maxent2 DO
    WHILE (J <= 4) AND (I <= Maxent2) DO
      IF (Tirun = TRUE) OR (Ecrun = TRUE)
      THEN
        Ibis := real(I);
        Hautint := Ibis * Echelle * 60.;
        Haut := entier(Hautint);
        Move(Rp^,82,195 - Haut);
        Draw(Rp^,85,195 - Haut);
        IF Ecrun = TRUE
        THEN Hautint := Hautint/8.;
```

```
          Hautint := Hautint + 0.5;
          Haut := entier(Hautint);
          wMove(Win2^,8,25 - Haut);
          ConvRealToString(Ibis,Istr,0,Decimal);
          PutStr(Win2^,ADR(Istr));
      END;
    END;
    J := J + 1;
    I := I + 1;
END;
IF (I <= Maxent2)
THEN
    IF (Tircinq = TRUE) OR (Ecrcinq = TRUE)
    THEN
      Ibis := real(I);
      Hautint := Ibis * Echelle * 60.;
      Haut := entier(Hautint);
      Move(Rp^,80,195 - Haut);
      Draw(Rp^,85,195 - Haut);
      IF Ecrcinq = TRUE
      THEN Hautint := Hautint/8.;
          Hautint := Hautint + 0.5;
          Haut := entier(Hautint);
          wMove(Win2^,8,25 - Haut);
          ConvRealToString(Ibis,Istr,0,Decimal);
          PutStr(Win2^,ADR(Istr));
      END;
    END;
    J := J + 1;
    I := I + 1;
END;
WHILE (J <= 9) AND (I <= Maxent2) DO
    IF (Tirun = TRUE) OR (Ecrun = TRUE)
    THEN
      Ibis := real(I);
      Hautint := Ibis * Echelle * 60.;
      Haut := entier(Hautint);
      Move(Rp^,82,195 - Haut);
      Draw(Rp^,85,195 - Haut);
      IF Ecrun = TRUE
      THEN Hautint := Hautint/8.;
          Hautint := Hautint + 0.5;
          Haut := entier(Hautint);
          wMove(Win2^,8,25 - Haut);
          ConvRealToString(Ibis,Istr,0,Decimal);
          PutStr(Win2^,ADR(Istr));
      END;
    END;
    J := J + 1;
    I := I + 1;
END;
IF (I <= Maxent2)
THEN
      Ibis := real(I);
      Hautint := Ibis * Echelle * 60.;
      Haut := entier(Hautint);
      Move(Rp^,80,195 - Haut);
      Draw(Rp^,85,195 - Haut);
      Hautint := Hautint/8.;
      Hautint := Hautint + 0.5;
      Haut := entier(Hautint);
      wMove(Win2^,8,25 - Haut);
      ConvRealToString(Ibis,Istr,0,Decimal);
      PutStr(Win2^,ADR(Istr));
      J := 1;
      I := I + 1;
```

```
    END;
END;
IF Tirquinze = TRUE
THEN
 J := 1;
 I := 1;
 WHILE I <= Maxent DO
    WHILE (J <= 3) AND (I <= Maxent) DO
        Ibis := real(I);
        Hautint := Ibis * Echelle * 15.;
        Haut := entier(Hautint);
        Move(Rp^,82,195 - Haut);
        Draw(Rp^,85,195 - Haut);
        J := J + 1;
        I := I + 1;
    END;
    J := 1;
    I := I + 1;
 END;
END;

ELSE
    I := 1;
    WHILE (I<=Maxent) DO
        Ibis := real(I);
        Hautint := Ibis * Echelle * 15.;
        Haut := entier(Hautint);
        Move(Rp^,82,195 - Haut);
        Draw(Rp^,85,195 - Haut);
        Hautint := Hautint/8.;
        Hautint := Hautint + 0.5;
        Haut := entier(Hautint);
        wMove(Win2^,8,25 - Haut);
        Ibis := Ibis * 15. ;
        ConvRealToString(Ibis,Istr,0,Decimal);
        PutStr(Win2^,ADR(Istr));
        I := I + 1;
    END;
END;
I := Indse;
Abs := 85       ;
WHILE (I <= DV) AND (I <= Fin) DO
    Move(Rp^,Abs,198);
    Draw(Rp^,Abs,195);
    Absre := real(Abs);
    Absre := Absre/8.;
    Absre := Absre + 0.5;
    Abstxt := entier(Absre);
    wMove(Win2^,Abstxt,26);
    Ibis := real(I+1);
    ConvRealToString(Ibis,Istr,0,Decimal);
    PutStr(Win2^,ADR(Istr));
    I := I + 1;
    Abs := Abs + Dist;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,2);
IF (TempsSommeInit[I] # 0)
THEN
  Ordint := real(TempsSommeInit[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
```

```
  en := WritePixel(Rp^,Abs - 1,195-Ord);
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
    IF Blanc = FALSE
    THEN IF TempsSommeInit[I] # 0
         THEN Ordint := real(TempsSommeInit[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Draw(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
         ELSE Blanc := TRUE;
         END;
    ELSE IF TempsSommeInit[I] # 0
         THEN Ordint := real(TempsSommeInit[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Move(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
              IF (I<DV)
              THEN
                IF (TempsSommeInit[I+1] = 0)
                THEN
                  en := WritePixel(Rp^,Abs - 1,195 - Ord);
                  en := WritePixel(Rp^,Abs + 1,195 - Ord);
                  en := WritePixel(Rp^,Abs ,194 - Ord);
                  en := WritePixel(Rp^,Abs ,196 - Ord);
                END;
              ELSE
                en := WritePixel(Rp^,Abs - 1,195 - Ord);
                en := WritePixel(Rp^,Abs + 1,195 - Ord);
                en := WritePixel(Rp^,Abs ,194 - Ord);
                en := WritePixel(Rp^,Abs ,196 - Ord);
              END;
              Blanc := FALSE;
          END;
    END;
    Abs := Abs + Dist;
    I := I + 1;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,3);
IF (TempsSommeRec[I] # 0)
THEN
  Ordint := real(TempsSommeRec[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
  en := WritePixel(Rp^,Abs - 1,195-Ord);
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
    IF Blanc = FALSE
    THEN IF TempsSommeRec[I] # 0
         THEN Ordint := real(TempsSommeRec[I]);
```

```
                Ordint := Ordint * Echelle;
                Ord := entier(Ordint);
                Draw(Rp^,Abs,195 - Ord);
                en := WritePixel(Rp^,Abs,195 - Ord);
           ELSE Blanc := TRUE;
           END;
      ELSE IF TempsSommeRec[I] # 0
           THEN Ordint := real(TempsSommeRec[I]);
                Ordint := Ordint * Echelle;
                Ord := entier(Ordint);
                Move(Rp^,Abs,195 - Ord);
                en := WritePixel(Rp^,Abs,195 - Ord);
                IF (I<DV)
                THEN
                  IF (TempsSommeRec[I+1] = 0)
                  THEN
                    en := WritePixel(Rp^,Abs - 1,195 - Ord);
                    en := WritePixel(Rp^,Abs + 1,195 - Ord);
                    en := WritePixel(Rp^,Abs ,194 - Ord);
                    en := WritePixel(Rp^,Abs ,196 - Ord);
                  END;
                ELSE
                  en := WritePixel(Rp^,Abs - 1,195 - Ord);
                  en := WritePixel(Rp^,Abs + 1,195 - Ord);
                  en := WritePixel(Rp^,Abs ,194 - Ord);
                  en := WritePixel(Rp^,Abs ,196 - Ord);
                END;

                Blanc := FALSE;
            END;
      END;
      Abs := Abs + Dist;
      I := I + 1;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,4);
IF (TempsSommeDep[I] # 0)
THEN
  Ordint := real(TempsSommeDep[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
  en := WritePixel(Rp^,Abs - 1,195-Ord);
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
      IF Blanc = FALSE
      THEN IF TempsSommeDep[I] # 0
           THEN Ordint := real(TempsSommeDep[I]);
                Ordint := Ordint * Echelle;
                Ord := entier(Ordint);
                Draw(Rp^,Abs,195 - Ord);
                en := WritePixel(Rp^,Abs,195 - Ord);
           ELSE Blanc := TRUE;
           END;
      ELSE IF TempsSommeDep[I] # 0
           THEN Ordint := real(TempsSommeDep[I]);
                Ordint := Ordint * Echelle;
                Ord := entier(Ordint);
```

```
                Move(Rp^,Abs,195 - Ord);
                en := WritePixel(Rp^,Abs,195 - Ord);
                IF (I<DV)
                THEN
                  IF (TempsSommeDep[I+1] = 0)
                  THEN
                    en := WritePixel(Rp^,Abs - 1,195 - Ord);
                    en := WritePixel(Rp^,Abs + 1,195 - Ord);
                    en := WritePixel(Rp^,Abs ,194 - Ord);
                    en := WritePixel(Rp^,Abs ,196 - Ord);
                  END;
                ELSE
                  en := WritePixel(Rp^,Abs - 1,195 - Ord);
                  en := WritePixel(Rp^,Abs + 1,195 - Ord);
                  en := WritePixel(Rp^,Abs ,194 - Ord);
                  en := WritePixel(Rp^,Abs ,196 - Ord);
                END;

                Blanc := FALSE;
            END;
    END;
    Abs := Abs + Dist;
    I := I + 1;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,6);
IF (TempsSIBillets[I] # 0)
THEN
  Ordint := real(TempsSIBillets[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs - 1,195-Ord);
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
    IF Blanc = FALSE
    THEN IF TempsSIBillets[I] # 0
         THEN Ordint := real(TempsSIBillets[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Draw(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
         ELSE Blanc := TRUE;
         END;
    ELSE IF TempsSIBillets[I] # 0
         THEN Ordint := real(TempsSIBillets[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Move(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
              IF (I<DV)
              THEN
                IF (TempsSIBillets[I+1] = 0)
                THEN
                  en := WritePixel(Rp^,Abs - 1,195 - Ord);
                  en := WritePixel(Rp^,Abs + 1,195 - Ord);
                  en := WritePixel(Rp^,Abs ,194 - Ord);
                  en := WritePixel(Rp^,Abs ,196 - Ord);
```

```
                END;
              ELSE
                en := WritePixel(Rp^,Abs - 1,195 - Ord);
                en := WritePixel(Rp^,Abs + 1,195 - Ord);
                en := WritePixel(Rp^,Abs ,194 - Ord);
                en := WritePixel(Rp^,Abs ,196 - Ord);
              END;


              Blanc := FALSE;
          END;
    END;
    Abs := Abs + Dist;
    I := I + 1;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,5);
IF (TempsCorrection[I] # 0)
THEN
  Ordint := real(TempsCorrection[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs - 1,195-Ord);
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
    IF Blanc = FALSE
    THEN IF TempsCorrection[I] # 0
         THEN Ordint := real(TempsCorrection[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Draw(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
         ELSE Blanc := TRUE;
         END;
    ELSE IF TempsCorrection[I] # 0
         THEN Ordint := real(TempsCorrection[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Move(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
              IF (I<DV)
              THEN
                IF (TempsCorrection[I+1] = 0)
                THEN
                  en := WritePixel(Rp^,Abs - 1,195 - Ord);
                  en := WritePixel(Rp^,Abs + 1,195 - Ord);
                  en := WritePixel(Rp^,Abs ,194 - Ord);
                  en := WritePixel(Rp^,Abs ,196 - Ord);
                END;
              ELSE
                en := WritePixel(Rp^,Abs - 1,195 - Ord);
                en := WritePixel(Rp^,Abs + 1,195 - Ord);
                en := WritePixel(Rp^,Abs ,194 - Ord);
                en := WritePixel(Rp^,Abs ,196 - Ord);
              END;
```

```
                Blanc := FALSE;
            END;
    END;
    Abs := Abs + Dist;
    I := I + 1;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,7);
IF (TempsMomentRec[I] # 0)
THEN
  Ordint := real(TempsMomentRec[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
  en := WritePixel(Rp^,Abs - 1,195-Ord);
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
    IF Blanc = FALSE
    THEN IF TempsMomentRec[I] # 0
         THEN Ordint := real(TempsMomentRec[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Draw(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
         ELSE Blanc := TRUE;
         END;
    ELSE IF TempsMomentRec[I] # 0
         THEN Ordint := real(TempsMomentRec[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Move(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
              IF (I<DV)
              THEN
                IF (TempsMomentRec[I+1] = 0)
                THEN
                  en := WritePixel(Rp^,Abs - 1,195 - Ord);
                  en := WritePixel(Rp^,Abs + 1,195 - Ord);
                  en := WritePixel(Rp^,Abs ,194 - Ord);
                  en := WritePixel(Rp^,Abs ,196 - Ord);
                END;
              ELSE
                en := WritePixel(Rp^,Abs - 1,195 - Ord);
                en := WritePixel(Rp^,Abs + 1,195 - Ord);
                en := WritePixel(Rp^,Abs ,194 - Ord);
                en := WritePixel(Rp^,Abs ,196 - Ord);
              END;


              Blanc := FALSE;
         END;
    END;
    Abs := Abs + Dist;
    I := I + 1;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,8);
```

```
IF (TempsMomentDep[I] # 0)
THEN
  Ordint := real(TempsMomentDep[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs - 1,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
    IF Blanc = FALSE
    THEN IF TempsMomentDep[I] # 0
         THEN Ordint := real(TempsMomentDep[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Draw(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
         ELSE Blanc := TRUE;
         END;
    ELSE IF TempsMomentDep[I] # 0
         THEN Ordint := real(TempsMomentDep[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Move(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
              IF (I<DV)
              THEN
                IF (TempsMomentDep[I+1] = 0)
                THEN
                  en := WritePixel(Rp^,Abs - 1,195 - Ord);
                  en := WritePixel(Rp^,Abs + 1,195 - Ord);
                  en := WritePixel(Rp^,Abs ,194 - Ord);
                  en := WritePixel(Rp^,Abs ,196 - Ord);
                END;
              ELSE
                en := WritePixel(Rp^,Abs - 1,195 - Ord);
                en := WritePixel(Rp^,Abs + 1,195 - Ord);
                en := WritePixel(Rp^,Abs ,194 - Ord);
                en := WritePixel(Rp^,Abs ,196 - Ord);
              END;


              Blanc := FALSE;
          END;
    END;
    Abs := Abs + Dist;
    I := I + 1;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,9);
IF (TempsPosteRec[I] # 0)
THEN
  Ordint := real(TempsPosteRec[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
  en := WritePixel(Rp^,Abs - 1,195-Ord);
```

```
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
    IF Blanc = FALSE
    THEN IF TempsPosteRec[I] # 0
         THEN Ordint := real(TempsPosteRec[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Draw(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
         ELSE Blanc := TRUE;
         END;
    ELSE IF TempsPosteRec[I] # 0
         THEN Ordint := real(TempsPosteRec[I]);
              Ordint := Ordint * Echelle;
              Ord := entier(Ordint);
              Move(Rp^,Abs,195 - Ord);
              en := WritePixel(Rp^,Abs,195 - Ord);
              IF (I<DV)
              THEN
                IF (TempsPosteRec[I+1] = 0)
                THEN
                  en := WritePixel(Rp^,Abs - 1,195 - Ord);
                  en := WritePixel(Rp^,Abs + 1,195 - Ord);
                  en := WritePixel(Rp^,Abs ,194 - Ord);
                  en := WritePixel(Rp^,Abs ,196 - Ord);
                END;
              ELSE
                en := WritePixel(Rp^,Abs - 1,195 - Ord);
                en := WritePixel(Rp^,Abs + 1,195 - Ord);
                en := WritePixel(Rp^,Abs ,194 - Ord);
                en := WritePixel(Rp^,Abs ,196 - Ord);
              END;


              Blanc := FALSE;
          END;
    END;
    Abs := Abs + Dist;
    I := I + 1;
END;
I := Indse;
Abs := 85;
SetAPen(Rp^,10);
IF (TempsPosteDep[I] # 0)
THEN
  Ordint := real(TempsPosteDep[I]);
  Ordint := Ordint* Echelle;
  Ord := entier(Ordint);
  Move(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs,195-Ord);
  en := WritePixel(Rp^,Abs - 1,195-Ord);
  en := WritePixel(Rp^,Abs + 1,195-Ord);
  Blanc := FALSE;
ELSE
  Blanc := TRUE;
END;
Abs := Abs + Dist;
I := I + 1;
WHILE (I <= DV) AND (I <= Fin) DO
    IF Blanc = FALSE
    THEN IF TempsPosteDep[I] # 0
```

```
                THEN Ordint := real(TempsPosteDep[I]);
                     Ordint := Ordint * Echelle;
                     Ord := entier(Ordint);
                     Draw(Rp^,Abs,195 - Ord);
                     en := WritePixel(Rp^,Abs,195 - Ord);
                ELSE Blanc := TRUE;
                END;
           ELSE IF TempsPosteDep[I] # 0
                THEN Ordint := real(TempsPosteDep[I]);
                     Ordint := Ordint * Echelle;
                     Ord := entier(Ordint);
                     Move(Rp^,Abs,195 - Ord);
                     en := WritePixel(Rp^,Abs,195 - Ord);
                     IF (I<DV)
                     THEN
                       IF (TempsPosteDep[I+1] = 0)
                       THEN
                         en := WritePixel(Rp^,Abs - 1,195 - Ord);
                         en := WritePixel(Rp^,Abs + 1,195 - Ord);
                         en := WritePixel(Rp^,Abs ,194 - Ord);
                         en := WritePixel(Rp^,Abs ,196 - Ord);
                       END;
                     ELSE
                       en := WritePixel(Rp^,Abs - 1,195 - Ord);
                       en := WritePixel(Rp^,Abs + 1,195 - Ord);
                       en := WritePixel(Rp^,Abs ,194 - Ord);
                       en := WritePixel(Rp^,Abs ,196 - Ord);
                     END;


                     Blanc := FALSE;
                END;
           END;
           Abs := Abs + Dist;
           I := I + 1;
      END;
      Indse := I;
      BeginGadgetList ();
      AddGadgetTextButton (1,1,ADR(" OK "));
      G14 := EndGadgetList ();
      Win3 := CreateWindow (312,226,56,12,WIDCMP,WFlags2,G14,Scr,NIL);
      Done4 := FALSE;
      WHILE (NOT Done4) DO
           Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
           LOOP
             Msg4 := GetMsg(Win3^.UserPort^);
             IF (Msg4 = NIL) THEN EXIT; END;
             ProcIMsg (Wp4, Msg4);
           END;
      END;
      CloseWindow(Win3^);
      FreeGadgetList (G14^);
      wClrScr (Win2^);
      END;
      DeleteConsole(Win2^);
     END;
     CloseWindow(Win2^);
    END;
   CloseScreen(Scr^);
   END;
END GraphTemps;

END Gtemps.
```

```
IMPLEMENTATION MODULE Transitions;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM Conversions IMPORT ConvNumberToString;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOS IMPORT IoErr,CreateDir,FileLock;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,ScreenToFront;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine;
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenOutputFile,
     OpenInputFile,CloseOutput,CloseInput,Done,ReadInt;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation, ConcatString,
     StringLength, ConvStringToUpperCase,OverwriteWithSubString;
FROM VariablesGlobales IMPORT Mattrans,Str3,Str9,Str20,Str40,TableauParametres,
                              nomuser,Numseance;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM Drawing IMPORT SetAPen,Move,Draw,WritePixel;
FROM RealInOut IMPORT WriteReal,ReadReal;
FROM RealConversions IMPORT ConvRealToString,RealToStringFormat,
     ConvStringToReal ;
FROM MathLib0 IMPORT entier,real;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Borderless};
  Blanc15 = "               ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
            END;

VAR
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Win,Win1,Win2,Win3,Win4 : WindowPtr;
  Indice,I, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  G1,G1OK,G12,G13,G14,G15 : GadgetPtr;
  Wp,Wp2,WpOK,Wp3,Wp4,WpImp : WindowProc;
  Sig : SignalSet;
```

```
  ReqOK,Req, Req2,Req3,Req5 : Requester;
  Msg,MsgOK,Msg2,Msg3,Msg4,Msg5 : IntuiMessagePtr;
  Done1,Done2,Done3,Done4,Efface,DoneOK,OK,Conf,UneSemaine,OKNombre,FenWin,
        PremPassage,Done5,Imp2 : BOOLEAN;
  Int : Gadget;
  Confirmation : Str40;
  Nom2,Max,NomInt,MaxInt : Str20;
  MAX : REAL;
  FileName : Str40;
  Fl : FileLock;
  le : CARDINAL;
  st : Str20;
  For : RealToStringFormat;
  len : INTEGER;
  lene : REAL;
  passe : Str20;
  reconnu : BOOLEAN;
  DV,essai : INTEGER;
  DoneNom : BOOLEAN;
  DernVers : REAL;
  LigneBlanc : ARRAY [0..77] OF CHAR;
  LigneSouligne : ARRAY [0..77] OF CHAR;
  imprime : BOOLEAN;
  Seance : Str20;

PROCEDURE Inittransitions (VAR DernVers : REAL;bo : BOOLEAN);

VAR I,J,Long,Valact,Valprec : INTEGER;
    St : Str20;
    Filechaine :Str40;
    Tabchaine : ARRAY [0..499] OF INTEGER;
    ent : INTEGER;
    ree,ree2 : REAL;
    OKSeance : BOOLEAN;

BEGIN
    Scr := CreateScreen (640,240,4,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 000FH;
      cmap[03] := 03F1H;
      cmap[04] := 0FE0H;
      cmap[05] := 0F0CH;
      cmap[06] := 00FFH;
      cmap[07] := 0870H;
      cmap[08] := 0E00H;
      cmap[09] := 0F90H;
      cmap[10] := 098FH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
```

```
cmap[28] := 0CCCH;
cmap[29] := 0DDDH;
cmap[30] := 0EEEH;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),16);
Win2 := CreateWindow (0,0,640,240,WIDCMP,WFlags,NIL,Scr,NIL);
IF (Win2 # NIL)
THEN IF CreateConsole(Win2^)
     THEN
      OKSeance := FALSE;
      WHILE (NOT OKSeance) DO
          wClrScr(Win2^);
          wMove(Win2^,30,2);
          IF bo
          THEN  PutStr(Win2^,ADR("TRANSITIONS CRITIQUES"));
                wMove(Win2^,30,3);
                PutStr(Win2^,ADR("----------------------"));
          ELSE  PutStr(Win2^,ADR("MATRICE DES TRANSITIONS"));
                wMove(Win2^,30,3);
                PutStr(Win2^,ADR("------------------------"));
          END;
          wMove(Win2^,7,10);
          PutStr(Win2^,ADR("Numéro de la séance [ de 1 à "));
          DernVers := DernVers + 1.;
          ConvRealToString(DernVers,St,0,Decimal);
          DernVers := DernVers - 1.;
          PutStr(Win2^,ADR(St));
          PutStr(Win2^,ADR(" ] : "));
          GetStr(Win2^,ADR(Numseance),SIZE(Numseance));
          ree := ConvStringToReal (Numseance);
          ent := entier (ree);
          ree2 := real(ent);
          IF (ree2 = ree) AND (ent > 0) AND (ree <= DernVers + 1.)
          THEN OKSeance := TRUE;
               ree := ree - 1.;
               ConvRealToString(ree,Seance,0,Decimal);
          END;
      END;
          Filechaine := "Param:";
          ConcatString(Filechaine,nomuser);
          ConcatString(Filechaine,Seance);
          ConcatString(Filechaine,".LONG_CHAINE");
          OpenInputFile(Filechaine);
          ScreenToFront(Scr^);
          IF Done
          THEN ReadInt(Long);
               Long := Long - 1;
               CloseInput;
               Filechaine := "Param:";
               ConcatString(Filechaine,nomuser);
               ConcatString(Filechaine,Seance);
               ConcatString(Filechaine,".CHAINE");
               OpenInputFile(Filechaine);
               ScreenToFront(Scr^);
               IF Done
               THEN I := 0;
                    WHILE (I <= Long) DO
                      ReadInt(Tabchaine[I]);
                      I := I + 1;
                    END;
                    CloseInput;
               END;
          END;

          I := 0;
          WHILE I <= 32 DO
```

```
                J := 0;
                WHILE J <= 32 DO
                  Mattrans[I,J] :=0;
                  J := J + 1;
                END;
                I := I + 1;
             END;

             Valprec := Tabchaine[0];
             I := 1;
             WHILE I <= Long DO
                Valact := Tabchaine[I];
                Mattrans[Valact,Valprec] := Mattrans[Valact,Valprec] + 1;
                Valprec := Tabchaine[I];
                I := I + 1;
             END;

             DeleteConsole(Win2^);
          END;
          CloseWindow(Win2^);
       END;
       CloseScreen(Scr^);
    END;
END Inittransitions;

PROCEDURE Ecriretitre (Num : ARRAY  OF CHAR);

BEGIN
   wClrScr(Win2^);
   wSetCursor (Win2^,FALSE);
   wMove(Win2^,30,2);
   PutStr(Win2^,ADR("Matrice des transitions [Partie "));
   PutStr(Win2^,ADR(Num));
   PutStr(Win2^,ADR(" ]"));
   wMove(Win2^,30,3);
   PutStr(Win2^,ADR("-----------------------------------"));
END Ecriretitre;

PROCEDURE EcrireLigneSup( Debabs,Finabs,Desdebabs : INTEGER);

VAR Ligne : ARRAY[0..77] OF CHAR;
    Indice,Debut,Fin : INTEGER;
    Re : REAL;
    St : Str20;

BEGIN
    CopyString(Ligne,LigneBlanc);
    OverwriteWithSubString(Ligne,"¦",Desdebabs - 2);
    Indice := Debabs;
    Debut := Desdebabs;
    WHILE Indice <= Finabs DO
       Re := real(Indice);
       ConvRealToString(Re,St,0,Decimal);
       OverwriteWithSubString(Ligne,St,Debut);
       Indice := Indice + 1;
       Debut := Debut + 4;
    END;
    PutStr(Win2^,ADR(Ligne));
    wMove(Win2^,1,7);
    CopyString(Ligne,LigneBlanc);
    Indice := Desdebabs - 6;
    Fin := Desdebabs + ((Finabs - Debabs + 1) * 4);
    WHILE Indice <= Fin DO
       OverwriteWithSubString(Ligne,"-",Indice);
       Indice := Indice + 1;
    END;
```

```
    PutStr(Win2^,ADR(Ligne));

END EcrireLigneSup;

PROCEDURE Dessinmat(Debabs : INTEGER; Finabs : INTEGER;
                    Debord : INTEGER; Finord : INTEGER;
                    Dabs : INTEGER;Dord : INTEGER);

VAR Ligne : ARRAY[0..77] OF CHAR;
    Ind,Inddes : INTEGER;
    Re : REAL;
    Stord,Stel : Str20;

BEGIN
    WHILE Debord <= Finord DO
      Ind := Debabs;
      Inddes := Dabs;
      CopyString(Ligne,LigneBlanc);
      Re := real(Debord);
      ConvRealToString(Re,Stord,0,Decimal);
      OverwriteWithSubString(Ligne,Stord,Inddes - 5);
      OverwriteWithSubString(Ligne,"¦",Inddes - 2);
      WHILE Ind <= Finabs DO
         Re := real(Mattrans[Ind,Debord]);
         ConvRealToString(Re,Stel,0,Decimal);
         OverwriteWithSubString(Ligne,Stel,Inddes);
         Ind := Ind + 1;
         Inddes := Inddes + 4;
      END;
      wMove(Win2^,1,Dord);
      PutStr(Win2^,ADR(Ligne));
      Debord := Debord + 1;
      Dord := Dord + 1;
   END;
END Dessinmat;


PROCEDURE JOUR (VAR StrDate:Str40);

VAR

  JJourStr,ANStr : Str20;
  DSR : DateStampRecord;
  MOIS,MOISPREC,Y : INTEGER;
  TrouveMois,TrouveAn : BOOLEAN;
  NBJours : ARRAY [0..11] OF CARDINAL;
  NomMois : ARRAY [0..11] OF Str9;
  Re,Re2,an : REAL;
  X : CARDINAL;
  Quatre : REAL;
  En : INTEGER;
  JJour,AN : INTEGER;

BEGIN

   DateStamp (DSR);
   X := SHORT (DSR.dsDays);
   Y := X;
   X := X + 1;
   AN := 78;
   TrouveAn := FALSE;
   NBJours [0] := 31;
   NBJours [2] := 31;
   NBJours [3] := 30;
   NBJours [4] := 31;
   NBJours [5] := 30;
```

```
NBJours [6] := 31;
NBJours [7] := 31;
NBJours [8] := 30;
NBJours [9] := 31;
NBJours [10] := 30;
NBJours [11] := 31;
NomMois [0] := "Janvier";
NomMois [1] := "Février";
NomMois [2] := "Mars";
NomMois [3] := "Avril";
NomMois [4] := "Mai";
NomMois [5] := "Juin";
NomMois [6] := "Juillet";
NomMois [7] := "Aout";
NomMois [8] := "Septembre";
NomMois [9] := "Octobre";
NomMois [10] := "Novembre";
NomMois [11] := "Décembre";
WHILE NOT TrouveAn DO
   an := real (AN);
   Quatre := 4.0;
   Re := an/Quatre;
   En := entier (Re);
   Re2 := real (En);
   IF Re=Re2
   THEN
      IF X>366
      THEN
         X := X - 366;
         AN := AN + 1;
      ELSE
         NBJours [1] := 29;
         TrouveAn := TRUE;
      END;
   ELSE
      IF X>365
      THEN
         X := X-365;
         AN := AN+1;
      ELSE
         NBJours [1]:=28;
         TrouveAn := TRUE;
      END;
   END;
END;
TrouveMois := FALSE;
MOISPREC := 11;
MOIS := 0;
WHILE NOT TrouveMois DO
   IF  X > NBJours [MOIS]
   THEN X := X - NBJours [MOIS];
        MOISPREC := MOIS;
        MOIS := MOIS +1;
   ELSE TrouveMois := TRUE;
   END;
END;
JJour := X;
CopyString (StrDate,"");
ConvNumberToString (JJourStr,LONGINT(JJour),FALSE,10,2," ");
ConcatString (StrDate,JJourStr);
ConcatString (StrDate," ");
ConcatString (StrDate,NomMois [MOIS]);
ConcatString (StrDate," ");
ConvNumberToString (ANStr,LONGINT(AN),FALSE,10,2," ");
ConcatString (StrDate,ANStr);
```

```
END JOUR;


PROCEDURE ImpLigneSup( Debabs,Finabs,Desdebabs : INTEGER);

VAR Ligne : ARRAY[0..77] OF CHAR;
    Indice,Debut,Fin : INTEGER;
    Re : REAL;
    St : Str20;

BEGIN
    CopyString(Ligne,LigneBlanc);
    OverwriteWithSubString(Ligne,"¦",Desdebabs - 2);
    Indice := Debabs;
    Debut := Desdebabs;
    WHILE Indice <= Finabs DO
       Re := real(Indice);
       ConvRealToString(Re,St,0,Decimal);
       OverwriteWithSubString(Ligne,St,Debut);
       Indice := Indice + 1;
       Debut := Debut + 4;
    END;
    WriteString(Ligne);
    WriteLn;
    CopyString(Ligne,LigneBlanc);
    Indice := Desdebabs - 6;
    Fin := Desdebabs + ((Finabs - Debabs + 1) * 4);
    WHILE Indice <= Fin DO
       OverwriteWithSubString(Ligne,"-",Indice);
       Indice := Indice + 1;
    END;
    WriteString(Ligne);

END ImpLigneSup;

PROCEDURE Impmat(Debabs : INTEGER; Finabs : INTEGER;
                 Debord : INTEGER; Finord : INTEGER;
                 Dabs : INTEGER);

VAR Ligne : ARRAY[0..77] OF CHAR;
    Ind,Inddes : INTEGER;
    Re : REAL;
    Stord,Stel : Str20;

BEGIN
    WHILE Debord <= Finord DO
      CloseOutput ();
      OpenOutputFile ("prt:");
      Ind := Debabs;
      Inddes := Dabs;
      CopyString(Ligne,LigneBlanc);
      Re := real(Debord);
      ConvRealToString(Re,Stord,0,Decimal);
      OverwriteWithSubString(Ligne,Stord,Inddes - 5);
      OverwriteWithSubString(Ligne,"¦",Inddes - 2);
      WHILE Ind <= Finabs DO
         Re := real(Mattrans[Ind,Debord]);
         ConvRealToString(Re,Stel,0,Decimal);
         OverwriteWithSubString(Ligne,Stel,Inddes);
         Ind := Ind + 1;
         Inddes := Inddes + 4;
      END;
      WriteLn;
      WriteString(Ligne);
      Debord := Debord + 1;
```

```
    END;
END Impmat;

PROCEDURE GadgetHandlerImp (VAR w:Window; VAR Msg5 : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF

    0 : |
    1 : |
    2 : Done5 := TRUE; Imp2 := TRUE |
    3 : Done5 := TRUE;
  END;

END GadgetHandlerImp;


PROCEDURE Imptransitions ;

VAR StTemps : Str40;

BEGIN

    Imp2 := FALSE;
    WITH WpImp DO
      procGadgetUp := GadgetHandlerImp;
    END;
    BeginGadgetList();
    GadgetTypeReq := TRUE;
    GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                        RelVerify});
    AddGadgetTextButton (60,10,ADR(" L'imprimante est allumée "));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
    GadgetMutualExcludeSet {});
    AddGadgetTextButton (110,30,ADR(" et 'ON-LINE' ? "));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
    GadgetMutualExcludeSet {});
    AddGadgetTextButton (110,50,ADR(" OUI "));
    AddGadgetTextButton (195,50,ADR(" NON "));
    G15 := EndGadgetList();
    InitRequester (Req5);
    WITH Req5 DO
            OlderRequest := NIL;
            LeftEdge := 150;
            TopEdge := 140;
            Width := 340;
            Height := 80;
            ReqGadget := G15;
            ReqBorder := NIL;
            ReqText := NIL;
            Flags := RequesterFlagsSet {};
            BackFill := BYTE(8);
            ReqLayer := NIL;
            ImageBMap := NIL;
    END;


    IF Request (Req5,Win2^)
    THEN
    Done5 := FALSE;
    WHILE (NOT Done5) DO
          Sig := Wait(SignalSet{CARDINAL(Win2^.UserPort^.mpSigBit)});
          LOOP
            Msg5 := GetMsg(Win2^.UserPort^);
            IF (Msg5 = NIL) THEN EXIT; END;
```

```
          ProcIMsg (WpImp, Msg5);
        END;
 END;
 END;
IF Imp2 = TRUE
THEN

 OpenOutputFile ("prt:");
 IF Done
 THEN
    WriteLn;
    WriteLn;
    WriteLn;
    WriteString("Nom de l'utilisateur : ");
    WriteString(nomuser);
    WriteLn;
    WriteLn;
    WriteString("Numéro de la séance : ");
    WriteString(Numseance);
    WriteLn;
    WriteLn;
    StTemps :="";
    JOUR ( StTemps);
    WriteString("Date de l'impression : ");
    WriteString(StTemps);
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteString("                                MATRICE DES TRANSITIONS");
    WriteLn;
    WriteString("                                ------------------------");
    WriteLn;
    WriteLn;
    CloseOutput ();
    OpenOutputFile ("prt:");
    ImpLigneSup (0,15,10);
    Impmat (0,15,0,32,10);

    CloseOutput ();
    Delay (1000);
    OpenOutputFile ("prt:");
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    CloseOutput ();
    OpenOutputFile ("prt:");
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
```

```
        WriteLn;
        CloseOutput ();
        OpenOutputFile ("prt:");
        ImpLigneSup (16,32,10);
        Impmat (16,32,0,32,10);
        WriteLn;
        WriteLn;
        WriteLn;
        WriteLn;
        WriteLn;
        WriteLn;

     END;
     CloseOutput ();
    END;

    FreeGadgetList (G15^);
END Imptransitions;


PROCEDURE GadgetHandler4 (VAR W:Window; VAR Msg4 : MsgData; VAR Gad : Gadget);

BEGIN

    Done4 := TRUE;

    IF Gad.GadgetID = 1
    THEN imprime := TRUE;
    END;

END GadgetHandler4;


PROCEDURE Saisieok (impr : BOOLEAN);

BEGIN
       BeginGadgetList ();
       AddGadgetTextButton (1,1,ADR(" OK "));
       IF impr = TRUE
       THEN  AddGadgetTextButton (110,1,ADR(" IMPRESSION "));
       END;
       G14 := EndGadgetList ();
       Win3 := CreateWindow (312,226,320,12,WIDCMP,WFlags2,G14,Scr,NIL);
       Done4 := FALSE;
       WHILE (NOT Done4) DO
           Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
           LOOP
             Msg4 := GetMsg(Win3^.UserPort^);
             IF (Msg4 = NIL) THEN EXIT; END;
             ProcIMsg (Wp4, Msg4);
           END;
       END;
       CloseWindow(Win3^);
       FreeGadgetList (G14^);

END Saisieok;



PROCEDURE Ecrantransitions ;

VAR Num : ARRAY [0..0] OF CHAR;

BEGIN

    WITH Wp4 DO
```

```
   procGadgetUp := GadgetHandler4;
END;

LigneBlanc := "                                                                ";
ConcatString(LigneBlanc,"                                  ");
LigneSouligne := "    ------------------------------------------";
ConcatString(LigneSouligne,"-------------------------  ");
Scr := CreateScreen (640,240,4,NIL);
IF (Scr # NIL) THEN
   cmap[00] := 0000H;
   cmap[01] := 0FFFH;
   cmap[02] := 000FH;
   cmap[03] := 03F1H;
   cmap[04] := 0FE0H;
   cmap[05] := 0F0CH;
   cmap[06] := 00FFH;
   cmap[07] := 0870H;
   cmap[08] := 0E00H;
   cmap[09] := 0F90H;
   cmap[10] := 098FH;
   cmap[11] := 007CH;
   cmap[12] := 000FH;
   cmap[13] := 070FH;
   cmap[14] := 0C0EH;
   cmap[15] := 0C08H;
   cmap[16] := 0620H;
   cmap[17] := 0E52H;
   cmap[18] := 0A52H;
   cmap[19] := 0FCAH;
   cmap[20] := 0333H;
   cmap[21] := 0444H;
   cmap[22] := 0555H;
   cmap[23] := 0666H;
   cmap[24] := 0777H;
   cmap[25] := 0888H;
   cmap[26] := 0999H;
   cmap[27] := 0AAAH;
   cmap[28] := 0CCCH;
   cmap[29] := 0DDDH;
   cmap[30] := 0EEEH;
   cmap[31] := 0FFFH;
   LoadRGB4 (Scr^.ViewPort,ADR (cmap),16);
   Win2 := CreateWindow (0,0,640,240,WIDCMP,WFlags,NIL,Scr,NIL);
   IF (Win2 # NIL)
   THEN IF CreateConsole(Win2^)
        THEN
             Ecriretitre("1");
             wMove(Win2^,1,6);
             EcrireLigneSup(0,15,12);
             Dessinmat(0,15,0,15,12,8);
             Saisieok(FALSE);
             Ecriretitre("2");
             wMove(Win2^,1,6);
             EcrireLigneSup(16,32,10);
             Dessinmat(16,32,0,15,10,8);
             Saisieok(FALSE);
             Ecriretitre("3");
             wMove(Win2^,1,6);
             EcrireLigneSup(0,15,12);
             Dessinmat(0,15,16,32,12,8);
             Saisieok(FALSE);
             Ecriretitre("4");
             wMove(Win2^,1,6);
             EcrireLigneSup(16,32,10);
             Dessinmat(16,32,16,32,10,8);
             imprime := FALSE;
```

```
                Saisieok(TRUE);
                IF (imprime = TRUE)
                THEN Imptransitions;
                END;
                DeleteConsole(Win2^);

            END;
            CloseWindow(Win2^);
        END;
        CloseScreen(Scr^);
    END;
END Ecrantransitions;

END Transitions.
```

```
IMPLEMENTATION MODULE temps;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM Conversions IMPORT ConvNumberToString;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOS IMPORT IoErr,CreateDir,FileLock;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,ScreenToFront;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine;
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenOutputFile,
     OpenInputFile,CloseOutput,CloseInput,Done,ReadInt;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation, ConcatString,
     StringLength, ConvStringToUpperCase,OverwriteWithSubString;
FROM VariablesGlobales IMPORT Str3,Str9, Str20, Str40, TableauParametres,
                              nomuser;
FROM Views IMPORT ViewPort,    adRGB4;
FROM Rasters IMPORT SetRast    astPortPtr;
FROM Drawing IMPORT SetAPen,Move,Draw,WritePixel;
FROM RealInOut IMPORT WriteReal,ReadReal;
FROM RealConversions IMPORT ConvRealToString,RealToStringFormat,
     ConvStringToReal ;
FROM MathLib0 IMPORT entier,real;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Borderless};
  Blanc15 = "               ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
            END;

VAR
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Win,Win1,Win2,Win3,Win4 : WindowPtr;
  Indice,I, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  G1,G1OK,G12,G13,G14,G15 : GadgetPtr;
  Wp,WpImp,Wp2,WpOK,Wp3,Wp4 : WindowProc;
  Sig : SignalSet;
```

```
  ReqOK,Req, Req2,Req3,Req5 : Requester;
  Msg,MsgOK,Msg2,Msg3,Msg4,Msg5 : IntuiMessagePtr;
  Done1,Done2,Done3,Done4,Efface,DoneOK,OK,Conf,UneSemaine,OKNombre,FenWin,
        PremPassage : BOOLEAN;
  Int : Gadget;
  Confirmation : Str40;
  Nom2,Max,NomInt,MaxInt : Str20;
  MAX : REAL;
  FileName : Str40;
  Fl : FileLock;
  le : CARDINAL;
  st : Str20;
  For : RealToStringFormat;
  len : INTEGER;
  lene : REAL;
  passe : Str20;
  reconnu : BOOLEAN;
  DV,essai : INTEGER;
  DoneNom : BOOLEAN;
  DernVers : REAL;
  TempsSommeInit,TempsSIBillets,TempsCorrection,TempsSommeRec,TempsMomentRec,
  TempsPosteRec,TempsSommeDep,TempsMomentDep,TempsPosteDep,
  TempsTotal : ARRAY [0..50] OF INTEGER;
  LigneBlanc : ARRAY [0..77] OF CHAR;
  LigneSouligne : ARRAY [0..77] OF CHAR;

  NTSI,NTSR,NTSD,NTCO,NTSB,NTMR,NTMD,NTPR,NTPD,NTT,MaxTSI,MinTSI,MaxTSR,
  MinTSR,MaxTSD,MinTSD,MaxTCO,MinTCO,MaxTSB,MinTSB,MaxTMR,MinTMR,MaxTMD,
  MinTMD,MaxTPR,MinTPR,MaxTPD,MinTPD,MaxTT,MinTT : INTEGER;
  imprime : BOOLEAN;
  Done5,Imp2 : BOOLEAN;

PROCEDURE JOUR (VAR StrDate:Str40);

VAR

  JJourStr,ANStr : Str20;
  DSR : DateStampRecord;
  MOIS,MOISPREC,Y : INTEGER;
  TrouveMois,TrouveAn : BOOLEAN;
  NBJours : ARRAY [0..11] OF CARDINAL;
  NomMois : ARRAY [0..11] OF Str9;
  Re,Re2,an : REAL;
  X : CARDINAL;
  Quatre : REAL;
  En : INTEGER;
  JJour,AN : INTEGER;

BEGIN

   DateStamp (DSR);
   X := SHORT (DSR.dsDays);
   Y := X;
   X := X + 1;
   AN := 78;
   TrouveAn := FALSE;
   NBJours [0] := 31;
   NBJours [2] := 31;
   NBJours [3] := 30;
   NBJours [4] := 31;
   NBJours [5] := 30;
   NBJours [6] := 31;
   NBJours [7] := 31;
   NBJours [8] := 30;
   NBJours [9] := 31;
   NBJours [10] := 30;
```

```
   NBJours [11] := 31;
   NomMois [0] := "Janvier";
   NomMois [1] := "Février";
   NomMois [2] := "Mars";
   NomMois [3] := "Avril";
   NomMois [4] := "Mai";
   NomMois [5] := "Juin";
   NomMois [6] := "Juillet";
   NomMois [7] := "Aout";
   NomMois [8] := "Septembre";
   NomMois [9] := "Octobre";
   NomMois [10] := "Novembre";
   NomMois [11] := "Décembre";
   WHILE NOT TrouveAn DO
      an := real (AN);
      Quatre := 4.0;
      Re := an/Quatre;
      En := entier (Re);
      Re2 := real (En);
      IF Re=Re2
      THEN
         IF X>366
         THEN
            X := X - 366;
            AN := AN + 1;
         ELSE
            NBJours [1] := 29;
            TrouveAn := TRUE;
         END;
      ELSE
         IF X>365
         THEN
            X := X-365;
            AN := AN+1;
         ELSE
            NBJours [1]:=28;
            TrouveAn := TRUE;
         END;
      END;
   END;
   TrouveMois := FALSE;
   MOISPREC := 11;
   MOIS := 0;
   WHILE NOT TrouveMois DO
      IF  X > NBJours [MOIS]
      THEN X := X - NBJours [MOIS];
           MOISPREC := MOIS;
           MOIS := MOIS +1;
      ELSE TrouveMois := TRUE;
      END;
   END;
   JJour := X;
   CopyString (StrDate,"");
   ConvNumberToString (JJourStr,LONGINT(JJour),FALSE,10,2," ");
   ConcatString (StrDate,JJourStr);
   ConcatString (StrDate," ");
   ConcatString (StrDate,NomMois [MOIS]);
   ConcatString (StrDate," ");
   ConvNumberToString (ANStr,LONGINT(AN),FALSE,10,2," ");
   ConcatString (StrDate,ANStr);

END JOUR;


PROCEDURE GadgetHandlerImp (VAR w:Window; VAR Msg5 : MsgData; VAR Gad: Gadget);
```

```
BEGIN

  CASE Gad.GadgetID OF

    0 : ¦
    1 : ¦
    2 : Done5 := TRUE; Imp2 := TRUE ¦
    3 : Done5 := TRUE;
  END;

END GadgetHandlerImp;



PROCEDURE ImpTemps (VAR DV,MaxTSI,MaxTSR,MaxTSD,MaxTSB,MaxTCO,
                    MaxTMR,MaxTMD,MaxTPR,MaxTPD,MaxTT:INTEGER ;
                    VAR TempsSommeInit,TempsSommeRec,TempsSommeDep,
                    TempsSIBillets,TempsCorrection,TempsMomentRec,
                    TempsMomentDep,TempsPosteRec,TempsPosteDep,
                    TempsTotal : ARRAY  OF INTEGER);

VAR

    Se : INTEGER;
    MTSI,MTSR,MTSD,MTCO,MTSB,MTMR,MTMD,MTPR,MTPD,MTT,REE,RE1 : REAL;
    StrTemps,StrSeance,StrMoy,StrMax,StrMin : Str20;
    Ligne,Ligne1,Ligne2,Ligne3 : ARRAY [0..77] OF CHAR;
    StTemps : Str40;

BEGIN

    Imp2 := FALSE;
    WITH WpImp DO
      procGadgetUp := GadgetHandlerImp;
    END;
    BeginGadgetList();
    GadgetTypeReq := TRUE;
    GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                        RelVerify});
    AddGadgetTextButton (60,10,ADR(" L'imprimante est allumée "));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
    GadgetMutualExcludeSet {});
    AddGadgetTextButton (110,30,ADR(" et 'ON-LINE' ? "));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
    GadgetMutualExcludeSet {});
    AddGadgetTextButton (110,50,ADR(" OUI "));
    AddGadgetTextButton (195,50,ADR(" NON "));
    G15 := EndGadgetList();
    InitRequester (Req5);
    WITH Req5 DO
            OlderRequest := NIL;
            LeftEdge := 150;
            TopEdge := 140;
            Width := 340;
            Height := 80;
            ReqGadget := G15;
            ReqBorder := NIL;
            ReqText := NIL;
            Flags := RequesterFlagsSet {};
            BackFill := BYTE(8);
            ReqLayer := NIL;
            ImageBMap := NIL;
    END;

    IF Request (Req5,Win2^)
    THEN
```

```
    Done5 := FALSE;
    WHILE (NOT Done5) DO
          Sig := Wait(SignalSet{CARDINAL(Win2^.UserPort^.mpSigBit)});
          LOOP
            Msg5 := GetMsg(Win2^.UserPort^);
            IF (Msg5 = NIL) THEN EXIT; END;
            ProcIMsg (WpImp, Msg5);
          END;
    END;
    END;
   IF Imp2 = TRUE
   THEN

    OpenOutputFile ("prt:");
    IF Done
    THEN
       WriteLn;
       WriteLn;
       WriteLn;
       WriteString("Nom de l'utilisateur : ");
       WriteString(nomuser);
       WriteLn;
       WriteLn;
       StTemps :="";
       JOUR ( StTemps);
       WriteString("Date de l'impression : ");
       WriteString(StTemps);
       WriteLn;
       WriteLn;
       WriteLn;
       WriteLn;
       WriteString("                                   Temps d'exécution ");
       WriteLn;
       WriteString("                                   ----------------- ");
       WriteLn;
       WriteLn;
       WriteString("- TSI = Temps nécessaire pour saisir la somme initiale");
       WriteLn;
       WriteString("- TSR = Temps nécessaire pour saisir le montant d'une rece
e");
       WriteLn;
       WriteString("- TSD = Temps nécessaire pour saisir le montant d'une dépe
e");
       WriteLn;
       WriteString("- TCO = Temps nécessaire pour corriger le montant initial"
       WriteLn;
       WriteString("- TSB = Temps nécessaire moyen pour cliquer sur un billet"
       WriteLn;
       WriteString("        lors de la saisie de la somme initiale");
       WriteLn;
       WriteString("- TMR = Temps nécessaire pour saisir le moment d'une recet
");
       WriteLn;
       WriteString("- TMD = Temps nécessaire pour saisir le moment d'une dépen
");
       WriteLn;
       WriteString("- TPR = Temps nécessaire pour saisir le poste d'une recett
);
       WriteLn;
       WriteString("- TPD = Temps nécessaire pour saisir le poste d'une dépens
);
       WriteLn;
       WriteString("- TT  = Temps total de la séance");
       WriteLn;
       WriteLn;
       WriteLn;
```

```
CloseOutput ();
OpenOutputFile("prt:");
CopyString(Ligne,LigneBlanc);
OverwriteWithSubString(Ligne,"¦ TSI",15);
OverwriteWithSubString(Ligne,"¦ TSR",21);
OverwriteWithSubString(Ligne,"¦ TSD",27);
OverwriteWithSubString(Ligne,"¦ TCO",33);
OverwriteWithSubString(Ligne,"¦ TSB",39);
OverwriteWithSubString(Ligne,"¦ TMR",45);
OverwriteWithSubString(Ligne,"¦ TMD",51);
OverwriteWithSubString(Ligne,"¦ TPR",57);
OverwriteWithSubString(Ligne,"¦ TPD",63);
OverwriteWithSubString(Ligne,"¦ TT",69);
WriteString(Ligne);
WriteLn;
WriteString(LigneSouligne);
Se := 0;
MTSI :=0.;
NTSI := 0;
MaxTSI :=0;
MinTSI :=10000;
MTSR :=0.;
NTSR := 0;
MaxTSR :=0;
MinTSR :=10000;
MTSD :=0.;
NTSD := 0;
MaxTSD :=0;
MinTSD :=10000;
MTCO :=0.;
NTCO := 0;
MaxTCO :=0;
MinTCO :=10000;
MTSB :=0.;
NTSB := 0;
MaxTSB :=0;
MinTSB :=10000;
MTMR :=0.;
NTMR := 0;
MaxTMR :=0;
MinTMR :=10000;
MTMD :=0.;
NTMD := 0;
MaxTMD :=0;
MinTMD :=10000;
MTPR :=0.;
NTPR := 0;
MaxTPR :=0;
MinTPR :=10000;
MTPD :=0.;
NTPD := 0;
MaxTPD :=0;
MinTPD :=10000;
MTT :=0.;
NTT := 0;
MaxTT :=0;
MinTT :=10000;
WriteLn;
WHILE (Se <= DV) DO
  CloseOutput ();
  OpenOutputFile("prt:");
  CopyString(Ligne,LigneBlanc);
  OverwriteWithSubString(Ligne,"Séance",5);
  REE := real(Se+1);
  ConvRealToString(REE,StrSeance,0,Decimal);
  OverwriteWithSubString(Ligne,StrSeance,12);
```

```
OverwriteWithSubString(Ligne,"¦",15);
OverwriteWithSubString(Ligne,"¦",15);
IF (TempsSommeInit[Se] # 0)
THEN
  RE1 := real(TempsSommeInit[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTSI := MTSI + RE1;
  NTSI := NTSI +1;
  IF (TempsSommeInit[Se] > MaxTSI)
  THEN MaxTSI := TempsSommeInit[Se];
  END;
  IF (TempsSommeInit[Se] < MinTSI)
  THEN MinTSI := TempsSommeInit[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,17);
ELSE
  OverwriteWithSubString(Ligne,"---",17);
END;
OverwriteWithSubString(Ligne,"¦",21);
IF (TempsSommeRec[Se] # 0)
THEN
  RE1 := real(TempsSommeRec[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTSR := MTSR + RE1;
  NTSR := NTSR +1;
  IF (TempsSommeRec[Se] > MaxTSR)
  THEN MaxTSR := TempsSommeRec[Se];
  END;
  IF (TempsSommeRec[Se] < MinTSR)
  THEN MinTSR := TempsSommeRec[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,23);
ELSE
  OverwriteWithSubString(Ligne,"---",23);
END;
OverwriteWithSubString(Ligne,"¦",27);
IF (TempsSommeDep[Se] # 0)
THEN
  RE1 := real(TempsSommeDep[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTSD := MTSD + RE1;
  NTSD := NTSD +1;
  IF (TempsSommeDep[Se] > MaxTSD)
  THEN MaxTSD := TempsSommeDep[Se];
  END;
  IF (TempsSommeDep[Se] < MinTSD)
  THEN MinTSD := TempsSommeDep[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,29);
ELSE
  OverwriteWithSubString(Ligne,"---",29);
END;
OverwriteWithSubString(Ligne,"¦",33);
IF (TempsCorrection[Se] # 0)
THEN
  RE1 := real(TempsCorrection[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTCO := MTCO + RE1;
  NTCO := NTCO +1;
  IF (TempsCorrection[Se] > MaxTCO)
  THEN MaxTCO := TempsCorrection[Se];
  END;
  IF (TempsCorrection[Se] < MinTCO)
  THEN MinTCO := TempsCorrection[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,35);
```

```
ELSE
  OverwriteWithSubString(Ligne,"---",35);
END;
OverwriteWithSubString(Ligne,"¦",39);
IF (TempsSIBillets[Se] # 0)
THEN
  RE1 := real(TempsSIBillets[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTSB := MTSB + RE1;
  NTSB := NTSB +1;
  IF (TempsSIBillets[Se] > MaxTSB)
  THEN MaxTSB := TempsSIBillets[Se];
  END;
  IF (TempsSIBillets[Se] < MinTSB)
  THEN MinTSB := TempsSIBillets[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,41);
ELSE
  OverwriteWithSubString(Ligne,"---",41);
END;
OverwriteWithSubString(Ligne,"¦",45);
IF (TempsMomentRec[Se] # 0)
THEN
  RE1 := real(TempsMomentRec[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTMR := MTMR + RE1;
  NTMR := NTMR +1;
  IF (TempsMomentRec[Se] > MaxTMR)
  THEN MaxTMR := TempsMomentRec[Se];
  END;
  IF (TempsMomentRec[Se] < MinTMR)
  THEN MinTMR := TempsMomentRec[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,47);
ELSE
  OverwriteWithSubString(Ligne,"---",47);
END;
OverwriteWithSubString(Ligne,"¦",51);
IF (TempsMomentDep[Se] # 0)
THEN
  RE1 := real(TempsMomentDep[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTMD := MTMD + RE1;
  NTMD := NTMD +1;
  IF (TempsMomentDep[Se] > MaxTMD)
  THEN MaxTMD := TempsMomentDep[Se];
  END;
  IF (TempsMomentDep[Se] < MinTMD)
  THEN MinTMD := TempsMomentDep[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,53);
ELSE
  OverwriteWithSubString(Ligne,"---",53);
END;
OverwriteWithSubString(Ligne,"¦",57);
IF (TempsPosteRec[Se] # 0)
THEN
  RE1 := real(TempsPosteRec[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTPR := MTPR + RE1;
  NTPR := NTPR +1;
  IF (TempsPosteRec[Se] > MaxTPR)
  THEN MaxTPR := TempsPosteRec[Se];
  END;
  IF (TempsPosteRec[Se] < MinTPR)
  THEN MinTPR := TempsPosteRec[Se];
```

```
    END;
    OverwriteWithSubString(Ligne,StrTemps,59);
  ELSE
    OverwriteWithSubString(Ligne,"---",59);
  END;
  OverwriteWithSubString(Ligne,"¦",63);
  IF (TempsPosteDep[Se] # 0)
  THEN
    RE1 := real(TempsPosteDep[Se]);
    ConvRealToString(RE1,StrTemps,0,Decimal);
    MTPD := MTPD + RE1;
    NTPD := NTPD +1;
    IF (TempsPosteDep[Se] > MaxTPD)
    THEN MaxTPD := TempsPosteDep[Se];
    END;
    IF (TempsPosteDep[Se] < MinTPD)
    THEN MinTPD := TempsPosteDep[Se];
    END;
    OverwriteWithSubString(Ligne,StrTemps,65);
  ELSE
    OverwriteWithSubString(Ligne,"---",65);
  END;
  OverwriteWithSubString(Ligne,"¦",69);
  IF (TempsTotal[Se] # 0)
  THEN
    RE1 := real(TempsTotal[Se]);
    ConvRealToString(RE1,StrTemps,0,Decimal);
    MTT := MTT + RE1;
    NTT := NTT +1;
    IF (TempsTotal[Se] > MaxTT)
    THEN MaxTT := TempsTotal[Se];
    END;
    IF (TempsTotal[Se] < MinTT)
    THEN MinTT := TempsTotal[Se];
    END;
    OverwriteWithSubString(Ligne,StrTemps,71);
  ELSE
    OverwriteWithSubString(Ligne,"---",71);
  END;
  WriteString(Ligne);
  WriteLn;
  Se := Se + 1;
END;
CloseOutput ();
OpenOutputFile ("prt:");
WriteString(LigneSouligne);
WriteLn;
CopyString(Ligne1,LigneBlanc);
CopyString(Ligne2,LigneBlanc);
CopyString(Ligne3,LigneBlanc);
OverwriteWithSubString(Ligne1,"Moyenne",5);
OverwriteWithSubString(Ligne2,"Maximum",5);
OverwriteWithSubString(Ligne3,"Minimum",5);
OverwriteWithSubString(Ligne1,"¦",15);
OverwriteWithSubString(Ligne2,"¦",15);
OverwriteWithSubString(Ligne3,"¦",15);
IF NTSI # 0
THEN
   RE1 := real(NTSI);
   MTSI := MTSI / RE1;
   ConvRealToString(MTSI,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,17);
   RE1 := real(MaxTSI);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,17);
   RE1 := real(MinTSI);
```

```
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,17);
ELSE
   OverwriteWithSubString(Ligne1,"---",17);
   OverwriteWithSubString(Ligne2,"---",17);
   OverwriteWithSubString(Ligne3,"---",17);
END;

OverwriteWithSubString(Ligne1,"¦",21);
OverwriteWithSubString(Ligne2,"¦",21);
OverwriteWithSubString(Ligne3,"¦",21);

IF NTSR # 0
THEN
   RE1 := real(NTSR);
   MTSR := MTSR / RE1;
   ConvRealToString(MTSR,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,23);
   RE1 := real(MaxTSR);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,23);
   RE1 := real(MinTSR);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,23);
ELSE
   OverwriteWithSubString(Ligne1,"---",23);
   OverwriteWithSubString(Ligne2,"---",23);
   OverwriteWithSubString(Ligne3,"---",23);
END;

OverwriteWithSubString(Ligne1,"¦",27);
OverwriteWithSubString(Ligne2,"¦",27);
OverwriteWithSubString(Ligne3,"¦",27);

IF NTSD # 0
THEN
   RE1 := real(NTSD);
   MTSD := MTSD / RE1;
   ConvRealToString(MTSD,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,29);
   RE1 := real(MaxTSD);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,29);
   RE1 := real(MinTSD);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,29);
ELSE
   OverwriteWithSubString(Ligne1,"---",29);
   OverwriteWithSubString(Ligne2,"---",29);
   OverwriteWithSubString(Ligne3,"---",29);
END;

OverwriteWithSubString(Ligne1,"¦",33);
OverwriteWithSubString(Ligne2,"¦",33);
OverwriteWithSubString(Ligne3,"¦",33);

IF NTCO # 0
THEN
   RE1 := real(NTCO);
   MTCO := MTCO / RE1;
   ConvRealToString(MTCO,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,35);
   RE1 := real(MaxTCO);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,35);
   RE1 := real(MinTCO);
```

```
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,35);
ELSE
   OverwriteWithSubString(Ligne1,"---",35);
   OverwriteWithSubString(Ligne2,"---",35);
   OverwriteWithSubString(Ligne3,"---",35);
END;

OverwriteWithSubString(Ligne1,"¦",39);
OverwriteWithSubString(Ligne2,"¦",39);
OverwriteWithSubString(Ligne3,"¦",39);

IF NTSB # 0
THEN
   RE1 := real(NTSB);
   MTSB := MTSB / RE1;
   ConvRealToString(MTSB,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,41);
   RE1 := real(MaxTSB);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,41);
   RE1 := real(MinTSB);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,41);
ELSE
   OverwriteWithSubString(Ligne1,"---",41);
   OverwriteWithSubString(Ligne2,"---",41);
   OverwriteWithSubString(Ligne3,"---",41);
END;


OverwriteWithSubString(Ligne1,"¦",45);
OverwriteWithSubString(Ligne2,"¦",45);
OverwriteWithSubString(Ligne3,"¦",45);

IF NTMR # 0
THEN
   RE1 := real(NTMR);
   MTMR := MTMR / RE1;
   ConvRealToString(MTMR,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,47);
   RE1 := real(MaxTMR);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,47);
   RE1 := real(MinTMR);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,47);
ELSE
   OverwriteWithSubString(Ligne1,"---",47);
   OverwriteWithSubString(Ligne2,"---",47);
   OverwriteWithSubString(Ligne3,"---",47);
END;

OverwriteWithSubString(Ligne1,"¦",51);
OverwriteWithSubString(Ligne2,"¦",51);
OverwriteWithSubString(Ligne3,"¦",51);

IF NTMD # 0
THEN
   RE1 := real(NTMD);
   MTMD := MTMD / RE1;
   ConvRealToString(MTMD,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,53);
   RE1 := real(MaxTMD);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,53);
```

```
   RE1 := real(MinTMD);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,53);
ELSE
   OverwriteWithSubString(Ligne1,"---",53);
   OverwriteWithSubString(Ligne2,"---",53);
   OverwriteWithSubString(Ligne3,"---",53);
END;

OverwriteWithSubString(Ligne1,"¦",57);
OverwriteWithSubString(Ligne2,"¦",57);
OverwriteWithSubString(Ligne3,"¦",57);

IF NTPR # 0
THEN
   RE1 := real(NTPR);
   MTPR := MTPR / RE1;
   ConvRealToString(MTPR,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,59);
   RE1 := real(MaxTPR);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,59);
   RE1 := real(MinTPR);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,59);
ELSE
   OverwriteWithSubString(Ligne1,"---",59);
   OverwriteWithSubString(Ligne2,"---",59);
   OverwriteWithSubString(Ligne3,"---",59);
END;

OverwriteWithSubString(Ligne1,"¦",63);
OverwriteWithSubString(Ligne2,"¦",63);
OverwriteWithSubString(Ligne3,"¦",63);

IF NTPD # 0
THEN
   RE1 := real(NTPD);
   MTPD := MTPD / RE1;
   ConvRealToString(MTPD,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,65);
   RE1 := real(MaxTPD);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,65);
   RE1 := real(MinTPD);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,65);
ELSE
   OverwriteWithSubString(Ligne1,"---",65);
   OverwriteWithSubString(Ligne2,"---",65);
   OverwriteWithSubString(Ligne3,"---",65);
END;

OverwriteWithSubString(Ligne1,"¦",69);
OverwriteWithSubString(Ligne2,"¦",69);
OverwriteWithSubString(Ligne3,"¦",69);

IF NTT # 0
THEN
   RE1 := real(NTT);
   MTT := MTT / RE1;
   ConvRealToString(MTT,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,71);
   RE1 := real(MaxTT);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,71);
```

```
         RE1 := real(MinTT);
         ConvRealToString(RE1,StrMin,0,Decimal);
         OverwriteWithSubString(Ligne3,StrMin,71);
      ELSE
         OverwriteWithSubString(Ligne1,"---",71);
         OverwriteWithSubString(Ligne2,"---",71);
         OverwriteWithSubString(Ligne3,"---",71);
      END;

      WriteString(Ligne1);
      WriteLn;
      WriteString(Ligne2);
      WriteLn;
      WriteString(Ligne3);
      WriteLn;
      WriteLn;
    END;
    WriteLn;
    WriteLn;
    WriteLn;
    CloseOutput ();
   END;

  FreeGadgetList(Gl5^);
END ImpTemps;



PROCEDURE InitTemps (VAR DV : INTEGER ;
                     VAR TempsSommeInit,TempsSommeRec,TempsSommeDep,
                     TempsSIBillets,TempsCorrection,TempsMomentRec,
                     TempsMomentDep,TempsPosteRec,TempsPosteDep,
                     TempsTotal : ARRAY  OF INTEGER);


VAR I : INTEGER;
    ree : REAL;
    res : Str40;
    StrIndice : Str20;

BEGIN
    I := 0;
    WHILE (I<=DV) DO
     TempsSommeInit[I] := 0;
     TempsSIBillets[I] := 0;
     TempsCorrection[I] := 0;
     TempsSommeRec[I] := 0;
     TempsMomentRec[I] := 0;
     TempsPosteRec [I] :=0;
     TempsSommeDep[I] := 0;
     TempsMomentDep[I]:=0;
     TempsPosteDep[I] :=0;
     TempsTotal[I] :=0;
     ree := real(I);
     ConvRealToString (ree,StrIndice,0,Decimal);
     res := "Param:";
     ConcatString(res,nomuser);
     ConcatString(res,StrIndice);
     ConcatString(res,".TEMPS");

     OpenInputFile (res);
     IF Done
     THEN
       ReadInt(TempsSommeInit[I]);
       ReadInt(TempsSIBillets[I]);
       ReadInt(TempsCorrection[I]);
```

```
        ReadInt(TempsSommeRec[I]);
        ReadInt(TempsMomentRec[I]);
        ReadInt(TempsPosteRec [I]);
        ReadInt(TempsSommeDep[I]);
        ReadInt(TempsMomentDep[I]);
        ReadInt(TempsPosteDep[I]);
        ReadInt(TempsTotal[I]);
        CloseInput;
      END;
      I := I+1;
    END;
END InitTemps;


PROCEDURE GadgetHandler4 (VAR W:Window; VAR Msg4 : MsgData; VAR Gad : Gadget);

BEGIN

   Done4 := TRUE;

   IF Gad.GadgetID = 1
   THEN imprime := TRUE;
   END;

END GadgetHandler4;


PROCEDURE Saisieok (impr : BOOLEAN);

BEGIN
      BeginGadgetList ();
      AddGadgetTextButton (1,1,ADR(" OK "));
      IF impr = TRUE
      THEN  AddGadgetTextButton (110,1,ADR(" IMPRESSION "));
      END;
      Gl4 := EndGadgetList ();
      Win3 := CreateWindow (312,226,320,12,WIDCMP,WFlags2,Gl4,Scr,NIL);
      Done4 := FALSE;
      WHILE (NOT Done4) DO
          Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
          LOOP
            Msg4 := GetMsg(Win3^.UserPort^);
            IF (Msg4 = NIL) THEN EXIT; END;
            ProcIMsg (Wp4, Msg4);
          END;
      END;
      CloseWindow(Win3^);
      FreeGadgetList (Gl4^);

END Saisieok;

PROCEDURE EcranTemps (VAR DV,MaxTSI,MaxTSR,MaxTSD,MaxTSB,
                      MaxTCO,MaxTMR,MaxTMD,MaxTPR,MaxTPD,MaxTT:INTEGER ;
                      VAR TempsSommeInit,TempsSommeRec,TempsSommeDep,
                      TempsSIBillets,TempsCorrection,TempsMomentRec,
                      TempsMomentDep,TempsPosteRec,TempsPosteDep,
                      TempsTotal : ARRAY  OF INTEGER);


VAR

    Se,Longecr : INTEGER;
    MTSI,MTSR,MTSD,MTCO,MTSB,MTMR,MTMD,MTPR,MTPD,MTT,REE,RE1 : REAL;
    Prem : BOOLEAN;
    StrTemps,StrSeance,StrMoy,StrMax,StrMin : Str20;
    Ligne,Ligne1,Ligne2,Ligne3 : ARRAY [0..77] OF CHAR;
```

```
BEGIN

    WITH Wp4 DO
       procGadgetUp := GadgetHandler4;
    END;

   LigneBlanc := "                                                    ";
   ConcatString(LigneBlanc,"                                  ");
   LigneSouligne := "     ------------------------------------------";
   ConcatString(LigneSouligne,"-------------------------  ");

    Scr := CreateScreen (640,240,4,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 000FH;
      cmap[03] := 03F1H;
      cmap[04] := 0FE0H;
      cmap[05] := 0F0CH;
      cmap[06] := 00FFH;
      cmap[07] := 0870H;
      cmap[08] := 0E00H;
      cmap[09] := 0F90H;
      cmap[10] := 098FH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),16);

    Win2:= CreateWindow(0,0,640,240,WIDCMP,WFlags2,NIL,Scr,NIL);
    IF (Win2 # NIL)
    THEN
     IF CreateConsole(Win2^)
     THEN
      wClrScr(Win2^);
      wSetCursor (Win2^,FALSE);
      wMove(Win2^,30,1);
      PutStr(Win2^,ADR("TEMPS D'EXECUTION"));
      wMove(Win2^,30,2);
      PutStr(Win2^,ADR("-----------------"));
      wMove(Win2^,5,4);
      PutStr(Win2^,ADR("Les écrans suivants vous présentent différents temps"));

      wMove(Win2^,3,10);
      PutStr(Win2^,ADR("- TSI = Temps nécessaire pour saisir la somme initiale'
);
      wMove(Win2^,3,11);
```

```
      PutStr(Win2^,ADR("- TSR = Temps nécessaire pour saisir le montant d'une re
cette"));
      wMove(Win2^,3,12);
      PutStr(Win2^,ADR("- TSD = Temps nécessaire pour saisir le montant d'une dé
pense"));
      wMove(Win2^,3,13);
      PutStr(Win2^,ADR("- TCO = Temps nécessaire pour corriger le montant initia
l"));
      wMove(Win2^,3,14);
      PutStr(Win2^,ADR("- TSB = Temps nécessaire pour cliquer sur un billet"));
      wMove(Win2^,3,15);
      PutStr(Win2^,ADR("        lors de la saisie de la somme initiale"));
      wMove(Win2^,3,16);
      PutStr(Win2^,ADR("- TMR = Temps nécessaire pour saisir le moment d'une rec
ette"));
      wMove(Win2^,3,17);
      PutStr(Win2^,ADR("- TMD = Temps nécessaire pour saisir le moment d'une dép
ense"));
      wMove(Win2^,3,18);
      PutStr(Win2^,ADR("- TPR = Temps nécessaire pour saisir le poste d'une rece
tte"));
      wMove(Win2^,3,19);
      PutStr(Win2^,ADR("- TPD = Temps nécessaire pour saisir le poste d'une dépe
nse"));
      wMove(Win2^,3,20);
      PutStr(Win2^,ADR("- TT  = Temps total de la séance"));
      DeleteConsole(Win2^);
    END;
    Saisieok (FALSE);
    IF CreateConsole(Win2^)
    THEN
      Se := 0;
      Prem := TRUE;
      MTSI :=0.;
      NTSI := 0;
      MaxTSI :=0;
      MinTSI :=10000;
      MTSR :=0.;
      NTSR := 0;
      MaxTSR :=0;
      MinTSR :=10000;
      MTSD :=0.;
      NTSD := 0;
      MaxTSD :=0;
      MinTSD :=10000;
      MTCO :=0.;
      NTCO := 0;
      MaxTCO :=0;
      MinTCO :=10000;
      MTSB :=0.;
      NTSB := 0;
      MaxTSB :=0;
      MinTSB :=10000;
      MTMR :=0.;
      NTMR := 0;
      MaxTMR :=0;
      MinTMR :=10000;
      MTMD :=0.;
      NTMD := 0;
      MaxTMD :=0;
      MinTMD :=10000;
      MTPR :=0.;
      NTPR := 0;
      MaxTPR :=0;
      MinTPR :=10000;
      MTPD :=0.;
```

```
NTPD := 0;
MaxTPD :=0;
MinTPD :=10000;
MTT :=0.;
NTT := 0;
MaxTT :=0;
MinTT :=10000;
WHILE (Se <= DV) DO
 IF Prem =FALSE
 THEN
   Saisieok (FALSE);
 END;
 wClrScr(Win2^);
 wSetCursor(Win2^,FALSE);
 CopyString(Ligne,LigneBlanc);
 OverwriteWithSubString(Ligne,"¦ TSI",15);
 OverwriteWithSubString(Ligne,"¦ TSR",21);
 OverwriteWithSubString(Ligne,"¦ TSD",27);
 OverwriteWithSubString(Ligne,"¦ TCO",33);
 OverwriteWithSubString(Ligne,"¦ TSB",39);
 OverwriteWithSubString(Ligne,"¦ TMR",45);
 OverwriteWithSubString(Ligne,"¦ TMD",51);
 OverwriteWithSubString(Ligne,"¦ TPR",57);
 OverwriteWithSubString(Ligne,"¦ TPD",63);
 OverwriteWithSubString(Ligne,"¦ TT",69);
 wMove(Win2^,1,2);
 PutStr(Win2^,ADR(Ligne));
 CopyString(Ligne,LigneSouligne);
 wMove(Win2^,1,3);
 PutStr(Win2^,ADR(Ligne));
 Longecr := 4;
 WHILE (Longecr<=19) AND (Se<=DV) DO
  CopyString(Ligne,LigneBlanc);
  OverwriteWithSubString(Ligne,"Séance",5);
  REE := real (Se+1);
  ConvRealToString(REE,StrSeance,0,Decimal);
  OverwriteWithSubString(Ligne,StrSeance,12);
  OverwriteWithSubString(Ligne,"¦",15);
  IF (TempsSommeInit[Se] # 0)
  THEN
    RE1 := real(TempsSommeInit[Se]);
    ConvRealToString(RE1,StrTemps,0,Decimal);
    MTSI := MTSI + RE1;
    NTSI := NTSI +1;
    IF (TempsSommeInit[Se] > MaxTSI)
    THEN MaxTSI := TempsSommeInit[Se];
    END;
    IF (TempsSommeInit[Se] < MinTSI)
    THEN MinTSI := TempsSommeInit[Se];
    END;
    OverwriteWithSubString(Ligne,StrTemps,17);
  ELSE
    OverwriteWithSubString(Ligne,"---",17);
  END;
  OverwriteWithSubString(Ligne,"¦",21);
  IF (TempsSommeRec[Se] # 0)
  THEN
    RE1 := real(TempsSommeRec[Se]);
    ConvRealToString(RE1,StrTemps,0,Decimal);
    MTSR := MTSR + RE1;
    NTSR := NTSR +1;
    IF (TempsSommeRec[Se] > MaxTSR)
    THEN MaxTSR := TempsSommeRec[Se];
    END;
    IF (TempsSommeRec[Se] < MinTSR)
    THEN MinTSR := TempsSommeRec[Se];
```

```
  END;
  OverwriteWithSubString(Ligne,StrTemps,23);
ELSE
  OverwriteWithSubString(Ligne,"---",23);
END;
OverwriteWithSubString(Ligne,"¦",27);
IF (TempsSommeDep[Se] # 0)
THEN
  RE1 := real(TempsSommeDep[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTSD := MTSD + RE1;
  NTSD := NTSD +1;
  IF (TempsSommeDep[Se] > MaxTSD)
  THEN MaxTSD := TempsSommeDep[Se];
  END;
  IF (TempsSommeDep[Se] < MinTSD)
  THEN MinTSD := TempsSommeDep[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,29);
ELSE
  OverwriteWithSubString(Ligne,"---",29);
END;
OverwriteWithSubString(Ligne,"¦",33);
IF (TempsCorrection[Se] # 0)
THEN
  RE1 := real(TempsCorrection[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTCO := MTCO + RE1;
  NTCO := NTCO +1;
  IF (TempsCorrection[Se] > MaxTCO)
  THEN MaxTCO := TempsCorrection[Se];
  END;
  IF (TempsCorrection[Se] < MinTCO)
  THEN MinTCO := TempsCorrection[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,35);
ELSE
  OverwriteWithSubString(Ligne,"---",35);
END;
OverwriteWithSubString(Ligne,"¦",39);
IF (TempsSIBillets[Se] # 0)
THEN
  RE1 := real(TempsSIBillets[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTSB := MTSB + RE1;
  NTSB := NTSB +1;
  IF (TempsSIBillets[Se] > MaxTSB)
  THEN MaxTSB := TempsSIBillets[Se];
  END;
  IF (TempsSIBillets[Se] < MinTSB)
  THEN MinTSB := TempsSIBillets[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,41);
ELSE
  OverwriteWithSubString(Ligne,"---",41);
END;
OverwriteWithSubString(Ligne,"¦",45);
IF (TempsMomentRec[Se] # 0)
THEN
  RE1 := real(TempsMomentRec[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTMR := MTMR + RE1;
  NTMR := NTMR +1;
  IF (TempsMomentRec[Se] > MaxTMR)
  THEN MaxTMR := TempsMomentRec[Se];
  END;
```

```
  IF (TempsMomentRec[Se] < MinTMR)
  THEN MinTMR := TempsMomentRec[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,47);
ELSE
  OverwriteWithSubString(Ligne,"---",47);
END;
OverwriteWithSubString(Ligne,"¦",51);
IF (TempsMomentDep[Se] # 0)
THEN
  RE1 := real(TempsMomentDep[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTMD := MTMD + RE1;
  NTMD := NTMD +1;
  IF (TempsMomentDep[Se] > MaxTMD)
  THEN MaxTMD := TempsMomentDep[Se];
  END;
  IF (TempsMomentDep[Se] < MinTMD)
  THEN MinTMD := TempsMomentDep[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,53);
ELSE
  OverwriteWithSubString(Ligne,"---",53);
END;
OverwriteWithSubString(Ligne,"¦",57);
IF (TempsPosteRec[Se] # 0)
THEN
  RE1 := real(TempsPosteRec[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTPR := MTPR + RE1;
  NTPR := NTPR +1;
  IF (TempsPosteRec[Se] > MaxTPR)
  THEN MaxTPR := TempsPosteRec[Se];
  END;
  IF (TempsPosteRec[Se] < MinTPR)
  THEN MinTPR := TempsPosteRec[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,59);
ELSE
  OverwriteWithSubString(Ligne,"---",59);
END;
OverwriteWithSubString(Ligne,"¦",63);
IF (TempsPosteDep[Se] # 0)
THEN
  RE1 := real(TempsPosteDep[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTPD := MTPD + RE1;
  NTPD := NTPD +1;
  IF (TempsPosteDep[Se] > MaxTPD)
  THEN MaxTPD := TempsPosteDep[Se];
  END;
  IF (TempsPosteDep[Se] < MinTPD)
  THEN MinTPD := TempsPosteDep[Se];
  END;
  OverwriteWithSubString(Ligne,StrTemps,65);
ELSE
  OverwriteWithSubString(Ligne,"---",65);
END;
OverwriteWithSubString(Ligne,"¦",69);
IF (TempsTotal[Se] # 0)
THEN
  RE1 := real(TempsTotal[Se]);
  ConvRealToString(RE1,StrTemps,0,Decimal);
  MTT := MTT + RE1;
  NTT := NTT +1;
  IF (TempsTotal[Se] > MaxTT)
```

```
    THEN MaxTT := TempsTotal[Se];
    END;
    IF (TempsTotal[Se] < MinTT)
    THEN MinTT := TempsTotal[Se];
    END;
    OverwriteWithSubString(Ligne,StrTemps,71);
  ELSE
    OverwriteWithSubString(Ligne,"---",71);
  END;
  wMove(Win2^,1,Longecr);
  PutStr(Win2^,ADR(Ligne));
  Se := Se + 1;
  Longecr := Longecr + 1;
 END;
 Prem := FALSE;
END;
CopyString(Ligne,LigneSouligne);
wMove (Win2^,1,Longecr);
PutStr(Win2^,ADR(Ligne));
wMove (Win2^,1,Longecr+1);
CopyString(Ligne1,LigneBlanc);
CopyString(Ligne2,LigneBlanc);
CopyString(Ligne3,LigneBlanc);
OverwriteWithSubString(Ligne1,"Moyenne",5);
OverwriteWithSubString(Ligne2,"Maximum",5);
OverwriteWithSubString(Ligne3,"Minimum",5);
OverwriteWithSubString(Ligne1,"¦",15);
OverwriteWithSubString(Ligne2,"¦",15);
OverwriteWithSubString(Ligne3,"¦",15);
IF NTSI # 0
THEN
   RE1 := real(NTSI);
   MTSI := MTSI / RE1;
   ConvRealToString(MTSI,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,17);
   RE1 := real(MaxTSI);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,17);
   RE1 := real(MinTSI);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,17);
ELSE
   OverwriteWithSubString(Ligne1,"---",17);
   OverwriteWithSubString(Ligne2,"---",17);
   OverwriteWithSubString(Ligne3,"---",17);
END;

OverwriteWithSubString(Ligne1,"¦",21);
OverwriteWithSubString(Ligne2,"¦",21);
OverwriteWithSubString(Ligne3,"¦",21);

IF NTSR # 0
THEN
   RE1 := real(NTSR);
   MTSR := MTSR / RE1;
   ConvRealToString(MTSR,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,23);
   RE1 := real(MaxTSR);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,23);
   RE1 := real(MinTSR);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,23);
ELSE
   OverwriteWithSubString(Ligne1,"---",23);
   OverwriteWithSubString(Ligne2,"---",23);
```

```
   OverwriteWithSubString(Ligne3,"---",23);
END;

OverwriteWithSubString(Ligne1,"¦",27);
OverwriteWithSubString(Ligne2,"¦",27);
OverwriteWithSubString(Ligne3,"¦",27);

IF NTSD # 0
THEN
   RE1 := real(NTSD);
   MTSD := MTSD / RE1;
   ConvRealToString(MTSD,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,29);
   RE1 := real(MaxTSD);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,29);
   RE1 := real(MinTSD);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,29);
ELSE
   OverwriteWithSubString(Ligne1,"---",29);
   OverwriteWithSubString(Ligne2,"---",29);
   OverwriteWithSubString(Ligne3,"---",29);
END;

OverwriteWithSubString(Ligne1,"¦",33);
OverwriteWithSubString(Ligne2,"¦",33);
OverwriteWithSubString(Ligne3,"¦",33);

IF NTCO # 0
THEN
   RE1 := real(NTCO);
   MTCO := MTCO / RE1;
   ConvRealToString(MTCO,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,35);
   RE1 := real(MaxTCO);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,35);
   RE1 := real(MinTCO);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,35);
ELSE
   OverwriteWithSubString(Ligne1,"---",35);
   OverwriteWithSubString(Ligne2,"---",35);
   OverwriteWithSubString(Ligne3,"---",35);
END;

OverwriteWithSubString(Ligne1,"¦",39);
OverwriteWithSubString(Ligne2,"¦",39);
OverwriteWithSubString(Ligne3,"¦",39);

IF NTSB # 0
THEN
   RE1 := real(NTSB);
   MTSB := MTSB / RE1;
   ConvRealToString(MTSB,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,41);
   RE1 := real(MaxTSB);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,41);
   RE1 := real(MinTSB);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,41);
ELSE
   OverwriteWithSubString(Ligne1,"---",41);
   OverwriteWithSubString(Ligne2,"---",41);
```

```
   OverwriteWithSubString(Ligne3,"---",41);
END;


OverwriteWithSubString(Ligne1,"¦",45);
OverwriteWithSubString(Ligne2,"¦",45);
OverwriteWithSubString(Ligne3,"¦",45);

IF NTMR # 0
THEN
   RE1 := real(NTMR);
   MTMR := MTMR / RE1;
   ConvRealToString(MTMR,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,47);
   RE1 := real(MaxTMR);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,47);
   RE1 := real(MinTMR);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,47);
ELSE
   OverwriteWithSubString(Ligne1,"---",47);
   OverwriteWithSubString(Ligne2,"---",47);
   OverwriteWithSubString(Ligne3,"---",47);
END;

OverwriteWithSubString(Ligne1,"¦",51);
OverwriteWithSubString(Ligne2,"¦",51);
OverwriteWithSubString(Ligne3,"¦",51);

IF NTMD # 0
THEN
   RE1 := real(NTMD);
   MTMD := MTMD / RE1;
   ConvRealToString(MTMD,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,53);
   RE1 := real(MaxTMD);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,53);
   RE1 := real(MinTMD);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,53);
ELSE
   OverwriteWithSubString(Ligne1,"---",53);
   OverwriteWithSubString(Ligne2,"---",53);
   OverwriteWithSubString(Ligne3,"---",53);
END;

OverwriteWithSubString(Ligne1,"¦",57);
OverwriteWithSubString(Ligne2,"¦",57);
OverwriteWithSubString(Ligne3,"¦",57);

IF NTPR # 0
THEN
   RE1 := real(NTPR);
   MTPR := MTPR / RE1;
   ConvRealToString(MTPR,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,59);
   RE1 := real(MaxTPR);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,59);
   RE1 := real(MinTPR);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,59);
ELSE
   OverwriteWithSubString(Ligne1,"---",59);
```

```
   OverwriteWithSubString(Ligne2,"---",59);
   OverwriteWithSubString(Ligne3,"---",59);
END;

OverwriteWithSubString(Ligne1,"¦",63);
OverwriteWithSubString(Ligne2,"¦",63);
OverwriteWithSubString(Ligne3,"¦",63);

IF NTPD # 0
THEN
   RE1 := real(NTPD);
   MTPD := MTPD / RE1;
   ConvRealToString(MTPD,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,65);
   RE1 := real(MaxTPD);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,65);
   RE1 := real(MinTPD);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,65);
ELSE
   OverwriteWithSubString(Ligne1,"---",65);
   OverwriteWithSubString(Ligne2,"---",65);
   OverwriteWithSubString(Ligne3,"---",65);
END;

OverwriteWithSubString(Ligne1,"¦",69);
OverwriteWithSubString(Ligne2,"¦",69);
OverwriteWithSubString(Ligne3,"¦",69);

IF NTT # 0
THEN
   RE1 := real(NTT);
   MTT := MTT / RE1;
   ConvRealToString(MTT,StrMoy,1,Decimal);
   OverwriteWithSubString(Ligne1,StrMoy,71);
   RE1 := real(MaxTT);
   ConvRealToString(RE1,StrMax,0,Decimal);
   OverwriteWithSubString(Ligne2,StrMax,71);
   RE1 := real(MinTT);
   ConvRealToString(RE1,StrMin,0,Decimal);
   OverwriteWithSubString(Ligne3,StrMin,71);
ELSE
   OverwriteWithSubString(Ligne1,"---",71);
   OverwriteWithSubString(Ligne2,"---",71);
   OverwriteWithSubString(Ligne3,"---",71);
END;

PutStr(Win2^,ADR(Ligne1));
wMove (Win2^,1,Longecr+2);
PutStr(Win2^,ADR(Ligne2));
wMove (Win2^,1,Longecr+3);
PutStr(Win2^,ADR(Ligne3));
wMove (Win2^,1,Longecr+4);

imprime := FALSE;
Saisieok(TRUE);
IF (imprime = TRUE)
THEN
          ImpTemps (DV,MaxTSI,MaxTSR,MaxTSD,MaxTSB,MaxTCO,MaxTMR,
                    MaxTMD,MaxTPR,MaxTPD,MaxTT,TempsSommeInit,
                    TempsSommeRec,TempsSommeDep,TempsSIBillets,
                    TempsCorrection,TempsMomentRec,TempsMomentDep,
                    TempsPosteRec,TempsPosteDep,TempsTotal);

END;
```

```
      DeleteConsole(Win2^);
     END;
     CloseWindow(Win2^);
    END;
   CloseScreen(Scr^);
   END;
END EcranTemps;


END temps.
```

```
IMPLEMENTATION MODULE Critiques;

FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM Conversions IMPORT ConvNumberToString;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOS IMPORT IoErr,CreateDir,FileLock;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,ScreenToFront;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine;
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenOutputFile,
    OpenInputFile,CloseOutput,CloseInput,Done,ReadInt;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation, ConcatString,
    StringLength, ConvStringToUpperCase,OverwriteWithSubString;
FROM VariablesGlobales IMPORT Mattrans,Str3,Str9,Str20,Str40,TableauParametres,
                              nomuser,Numseance;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM Drawing IMPORT SetAPen,Move,Draw,WritePixel;
FROM RealInOut IMPORT WriteReal,ReadReal;
FROM RealConversions IMPORT ConvRealToString,RealToStringFormat,
    ConvStringToReal ;
FROM MathLib0 IMPORT entier,real;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Borderless};
  Blanc15 = "               ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
            END;

VAR
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Win,Win1,Win2,Win3,Win4 : WindowPtr;
  Indice,I, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  G1,G1OK,G12,G13,G14,G15 : GadgetPtr;
  Wp,Wp2,WpOK,Wp3,Wp4,WpImp : WindowProc;
  Sig : SignalSet;
```

```
  ReqOK,Req, Req2,Req3,Req5 : Requester;
  Msg,MsgOK,Msg2,Msg3,Msg4,Msg5 : IntuiMessagePtr;
  Done1,Done2,Done3,Done4,Efface,DoneOK,OK,Conf,UneSemaine,OKNombre,FenWin,
        PremPassage,Done5,Imp2 : BOOLEAN;
  Int : Gadget;
  Confirmation : Str40;
  Nom2,Max,NomInt,MaxInt : Str20;
  MAX : REAL;
  FileName : Str40;
  Fl : FileLock;
  le : CARDINAL;
  st : Str20;
  For : RealToStringFormat;
  len : INTEGER;
  lene : REAL;
  passe : Str20;
  reconnu : BOOLEAN;
  DV,essai : INTEGER;
  DoneNom : BOOLEAN;
  DernVers : REAL;
  LigneBlanc : ARRAY [0..77] OF CHAR;
  Ligne : ARRAY [0..77] OF CHAR;
  imprime : BOOLEAN;
  Seance : Str20;


PROCEDURE JOUR (VAR StrDate:Str40);

VAR

  JJourStr,ANStr : Str20;
  DSR : DateStampRecord;
  MOIS,MOISPREC,Y : INTEGER;
  TrouveMois,TrouveAn : BOOLEAN;
  NBJours : ARRAY [0..11] OF CARDINAL;
  NomMois : ARRAY [0..11] OF Str9;
  Re,Re2,an : REAL;
  X : CARDINAL;
  Quatre : REAL;
  En : INTEGER;
  JJour,AN : INTEGER;

BEGIN

   DateStamp (DSR);
   X := SHORT (DSR.dsDays);
   Y := X;
   X := X + 1;
   AN := 78;
   TrouveAn := FALSE;
   NBJours [0] := 31;
   NBJours [2] := 31;
   NBJours [3] := 30;
   NBJours [4] := 31;
   NBJours [5] := 30;
   NBJours [6] := 31;
   NBJours [7] := 31;
   NBJours [8] := 30;
   NBJours [9] := 31;
   NBJours [10] := 30;
   NBJours [11] := 31;
   NomMois [0] := "Janvier";
   NomMois [1] := "Février";
   NomMois [2] := "Mars";
   NomMois [3] := "Avril";
   NomMois [4] := "Mai";
```

```
   NomMois [5] := "Juin";
   NomMois [6] := "Juillet";
   NomMois [7] := "Aout";
   NomMois [8] := "Septembre";
   NomMois [9] := "Octobre";
   NomMois [10] := "Novembre";
   NomMois [11] := "Décembre";
   WHILE NOT TrouveAn DO
      an := real (AN);
      Quatre := 4.0;
      Re := an/Quatre;
      En := entier (Re);
      Re2 := real (En);
      IF Re=Re2
      THEN
         IF X>366
         THEN
            X := X - 366;
            AN := AN + 1;
         ELSE
            NBJours [1] := 29;
            TrouveAn := TRUE;
         END;
      ELSE
         IF X>365
         THEN
            X := X-365;
            AN := AN+1;
         ELSE
            NBJours [1]:=28;
            TrouveAn := TRUE;
         END;
      END;
   END;
   TrouveMois := FALSE;
   MOISPREC := 11;
   MOIS := 0;
   WHILE NOT TrouveMois DO
      IF  X > NBJours [MOIS]
      THEN X := X - NBJours [MOIS];
           MOISPREC := MOIS;
           MOIS := MOIS +1;
      ELSE TrouveMois := TRUE;
      END;
   END;
   JJour := X;
   CopyString (StrDate,"");
   ConvNumberToString (JJourStr,LONGINT(JJour),FALSE,10,2," ");
   ConcatString (StrDate,JJourStr);
   ConcatString (StrDate," ");
   ConcatString (StrDate,NomMois [MOIS]);
   ConcatString (StrDate," ");
   ConvNumberToString (ANStr,LONGINT(AN),FALSE,10,2," ");
   ConcatString (StrDate,ANStr);

END JOUR;

PROCEDURE Calcultot ( ind: INTEGER; VAR Tot: INTEGER);

VAR I : INTEGER;

BEGIN
    I := 0;
    Tot := 0;
    WHILE (I <=32) DO
     Tot := Tot + Mattrans [I,ind];
```

```
     I := I + 1;
    END;

END Calcultot;

PROCEDURE Affligne (Phrase:ARRAY OF CHAR; l:INTEGER; i: INTEGER; j : INTEGER;
                    Tot: INTEGER );

VAR
    Interm :REAL;
    St : Str20;

BEGIN
    wMove(Win2^,2,1);
    PutStr(Win2^,ADR(Phrase));
    wMove(Win2^,70,1);
    PutStr(Win2^,ADR(": "));
    Interm := real(Mattrans [j,i]);
    ConvRealToString(Interm,St,0,Decimal);
    PutStr(Win2^,ADR(St));
    IF Tot # 0
    THEN PutStr(Win2^,ADR("/"));
         Interm := real(Tot);
         ConvRealToString(Interm,St,0,Decimal);
         PutStr(Win2^,ADR(St));
    END;
END Affligne;

PROCEDURE Impligne (Phrase:ARRAY OF CHAR; i: INTEGER; j : INTEGER;
                    Tot: INTEGER );

VAR
    Interm :REAL;
    St : Str20;

BEGIN

    CopyString (Ligne,LigneBlanc);
    OverwriteWithSubString (Ligne,Phrase,2);
    OverwriteWithSubString (Ligne,": ",70);
    Interm := real(Mattrans [j,i]);
    ConvRealToString(Interm,St,0,Decimal);
    OverwriteWithSubString(Ligne,St,72);
    IF Tot # 0
    THEN OverwriteWithSubString(Ligne,"/",74);
         Interm := real(Tot);
         ConvRealToString(Interm,St,0,Decimal);
         OverwriteWithSubString(Ligne,St,75);
    END;
    WriteString(Ligne);
    WriteLn;
END Impligne;

PROCEDURE GadgetHandlerImp (VAR w:Window; VAR Msg5 : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF

    0 : |
    1 : |
    2 : Done5 := TRUE; Imp2 := TRUE |
    3 : Done5 := TRUE;
  END;

END GadgetHandlerImp;
```

```
PROCEDURE Impcritiques ;

VAR StTemps : Str40;
    Tot : INTEGER;

BEGIN

    LigneBlanc := "                                                    ";
    ConcatString(LigneBlanc,"                                  ");

    Imp2 := FALSE;
    WITH WpImp DO
      procGadgetUp := GadgetHandlerImp;
    END;
    BeginGadgetList();
    GadgetTypeReq := TRUE;
    GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                     RelVerify});
    AddGadgetTextButton (60,10,ADR(" L'imprimante est allumée "));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
    GadgetMutualExcludeSet {});
    AddGadgetTextButton (110,30,ADR(" et 'ON-LINE' ? "));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
    GadgetMutualExcludeSet {});
    AddGadgetTextButton (110,50,ADR(" OUI "));
    AddGadgetTextButton (195,50,ADR(" NON "));
    G15 := EndGadgetList();
    InitRequester (Req5);
    WITH Req5 DO
            OlderRequest := NIL;
            LeftEdge := 150;
            TopEdge := 140;
            Width := 340;
            Height := 80;
            ReqGadget := G15;
            ReqBorder := NIL;
            ReqText := NIL;
            Flags := RequesterFlagsSet {};
            BackFill := BYTE(8);
            ReqLayer := NIL;
            ImageBMap := NIL;
    END;


    IF Request (Req5,Win2^)
    THEN
    Done5 := FALSE;
    WHILE (NOT Done5) DO
          Sig := Wait(SignalSet{CARDINAL(Win2^.UserPort^.mpSigBit)});
          LOOP
            Msg5 := GetMsg(Win2^.UserPort^);
            IF (Msg5 = NIL) THEN EXIT; END;
            ProcIMsg (WpImp, Msg5);
          END;
    END;
    END;
   IF Imp2 = TRUE
   THEN

    OpenOutputFile ("prt:");
    IF Done
    THEN
       WriteLn;
       WriteLn;
```

```
WriteString("Nom de l'utilisateur : ");
WriteString(nomuser);
WriteLn;
WriteLn;
WriteString("Numéro de la séance : ");
WriteString(Numseance);
WriteLn;
WriteLn;
StTemps :="";
JOUR ( StTemps);
WriteString("Date de l'impression : ");
WriteString(StTemps);
WriteLn;
WriteLn;
WriteLn;
WriteLn;
WriteString("                                TRANSITIONS CRITIQUES");
WriteLn;
WriteString("                                ---------------------");
WriteLn;
CloseOutput ();
OpenOutputFile ("prt:");
WriteLn;
WriteLn;
CopyString(Ligne,LigneBlanc);
OverwriteWithSubString(Ligne,"Lors de la saisie du nom : ",1);
WriteString(Ligne);
WriteLn;
WriteLn;
Calcultot (0,Tot);
Impligne ("- l'utilisateur désire recommencer la saisie du nom",
          0,1,Tot);
Calcultot (2,Tot);
Impligne ("- il infirme lorsqu'on lui demande si son nom est juste",
          2,0,Tot);
WriteLn;
CopyString(Ligne,LigneBlanc);
OverwriteWithSubString(Ligne,"Lors de la saisie de la somme initiale : "
             1);
WriteString (Ligne);
WriteLn;
WriteLn;
CloseOutput ();
OpenOutputFile ("prt:");
Calcultot (3,Tot);
Impligne ("- il demande d'effacer la dernière opération",3,4,Tot);
Impligne ("- il demande de recommencer la saisie",3,5,Tot);
Calcultot (6,Tot);
Impligne ("- il ne confirme pas la somme aprés avoir cliqué sur ok"
         ,6,3,Tot);

CopyString(Ligne,LigneBlanc);
WriteLn;
OverwriteWithSubString(Ligne,"Lors de la saisie d'une recette : ",
             1);
WriteString (Ligne);
WriteLn;
WriteLn;
CloseOutput ();
OpenOutputFile ("prt:");
Calcultot (9,Tot);
Impligne ("- il ne confirme pas le poste budgétaire qu'il a choisi",
          9,8,Tot);
Calcultot (11,Tot);
Impligne ("- il ne confirme pas le moment qu'il a choisi",
           11,10,Tot);
```

```
Calcultot (12,Tot);
Impligne ("- il demande d'effacer la dernière opération",
          12,13,Tot);
Impligne ("- il désire quitter la saisie",12,14,Tot);
Calcultot (14,Tot);
Impligne ("- il ne quitte pas aprés l'avoir demandé",14,12,Tot);
Impligne ("- il quitte réellement la saisie",14,7,Tot);
Calcultot (15,Tot);
CloseOutput ();
OpenOutputFile ("prt:");
Impligne ("- il ne confirme pas le montant après avoir cliqué sur ok",
           15,12,Tot);
WriteLn;
CloseOutput ();
OpenOutputFile ("prt:");
CopyString(Ligne,LigneBlanc);
OverwriteWithSubString(Ligne,"Lors de la saisie d'une dépense :",
              1);
WriteString (Ligne);
WriteLn;
WriteLn;
Calcultot (17,Tot);
Impligne ("- il ne confirme pas le poste budgétaire qu'il a choisi",
          17,16,Tot);
Calcultot (19,Tot);
Impligne ("- il ne confirme pas le moment qu'il a choisi",
          19,18,Tot);
Calcultot (20,Tot);
CloseOutput ();
OpenOutputFile ("prt:");
Impligne ("- il demande d'effacer la dernière opération",
          20,21,Tot);
Impligne ("- il désire quitter la saisie",20,22,Tot);
Calcultot (22,Tot);
Impligne ("- il ne quitte pas aprés l'avoir demandé",22,20,Tot);
Impligne ("- il quitte réellement la saisie",22,7,Tot);
Calcultot (23,Tot);
Impligne ("- il ne confirme pas le montant après avoir cliqué sur ok",
              23,20,Tot);

CopyString(Ligne,LigneBlanc);
WriteLn;
OverwriteWithSubString(Ligne,"Lors de la correction :",
              1);
CloseOutput ();
OpenOutputFile ("prt:");
WriteString (Ligne);
WriteLn;
WriteLn;
Calcultot (7,Tot);
IF Mattrans [7,25] # 0
 THEN  Impligne ("- il demande de corriger",7,25,Tot);
 ELSE  Impligne ("- il demande de corriger",7,26,Tot);
 END;
Calcultot (26,Tot);
Impligne ("- il demande d'effacer la dernière opération",26,27,Tot);
Impligne ("- il désire recommencer la saisie",26,28,Tot);
Calcultot (29,Tot);
Impligne ("- il ne confirme pas le montant aprés avoir cliqué sur ok",
           29,26,Tot);
CloseOutput ();
OpenOutputFile ("prt:");
CopyString(Ligne,LigneBlanc);
WriteLn;
OverwriteWithSubString(Ligne,"Lors de la fin du programme :",
              1);
```

```
      WriteString (Ligne);
      WriteLn;
      WriteLn;
      Calcultot (30,Tot);
      Impligne ("- il ne confirme pas sa demande de fin de session",
                30,7,Tot);
      Calcultot (31,Tot);
      Impligne ("- il revient au menu après présentation de la balance",
                31,7,Tot);

      WriteLn;
      WriteLn;
      WriteLn;
      CloseOutput ();
      OpenOutputFile ("prt:");
      WriteLn;

    END;
    CloseOutput ();
   END;

   FreeGadgetList (G15^);
END Impcritiques;


PROCEDURE GadgetHandler4 (VAR W:Window; VAR Msg4 : MsgData; VAR Gad : Gadget);

BEGIN

   Done4 := TRUE;

   IF Gad.GadgetID = 1
   THEN imprime := TRUE;
   END;

END GadgetHandler4;


PROCEDURE Saisieok (impr : BOOLEAN);

BEGIN
      BeginGadgetList ();
      AddGadgetTextButton (1,1,ADR(" OK "));
      IF impr = TRUE
      THEN  AddGadgetTextButton (110,1,ADR(" IMPRESSION "));
      END;
      G14 := EndGadgetList ();
      Win3 := CreateWindow (312,226,320,12,WIDCMP,WFlags2,G14,Scr,NIL);
      Done4 := FALSE;
      WHILE (NOT Done4) DO
          Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
          LOOP
            Msg4 := GetMsg(Win3^.UserPort^);
            IF (Msg4 = NIL) THEN EXIT; END;
            ProcIMsg (Wp4, Msg4);
          END;
      END;
      CloseWindow(Win3^);
      FreeGadgetList (G14^);

END Saisieok;



PROCEDURE Ecrancritiques ;
```

```
VAR Tot : INTEGER;


BEGIN

   WITH Wp4 DO
      procGadgetUp := GadgetHandler4;
   END;

   Scr := CreateScreen (640,240,4,NIL);
   IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 000FH;
      cmap[03] := 03F1H;
      cmap[04] := 0FE0H;
      cmap[05] := 0F0CH;
      cmap[06] := 00FFH;
      cmap[07] := 0870H;
      cmap[08] := 0E00H;
      cmap[09] := 0F90H;
      cmap[10] := 098FH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
      cmap[29] := 0DDDH;
      cmap[30] := 0EEEH;
      cmap[31] := 0FFFH;
      LoadRGB4 (Scr^.ViewPort,ADR (cmap),16);
      Win2 := CreateWindow (0,0,640,240,WIDCMP,WFlags,NIL,Scr,NIL);
      IF (Win2 # NIL)
      THEN
       IF CreateConsole(Win2^)
       THEN
        wSetCursor(Win2^,FALSE);
        wMove(Win2^,30,2);
        PutStr(Win2^,ADR("TRANSITIONS CRITIQUES"));
        wMove(Win2^,30,3);
        PutStr(Win2^,ADR("----------------------"));
        wMove(Win2^,1,6);
        PutStr(Win2^,ADR("Lors de la saisie du nom : "));
        Calcultot (0,Tot);
        Affligne ("- l'utilisateur désire recommencer la saisie du nom",
                  8,0,1,Tot);
        Calcultot (2,Tot);
        Affligne ("- il infirme lorsqu'on lui demande si son nom est juste",
                   9,2,0,Tot);
        wMove(Win2^,1,11);
        PutStr(Win2^,ADR("Lors de la saisie de la somme initiale : "));
        Calcultot (3,Tot);
        Affligne ("- il demande d'effacer la dernière opération",13,3,4,Tot);
```

```
Affligne ("- il demande de recommencer la saisie",14,3,5,Tot);
Calcultot (6,Tot);
Affligne ("- il ne confirme pas la somme aprés avoir cliqué sur ok"
        ,15,6,3,Tot);
wMove(Win2^,1,17);
PutStr(Win2^,ADR("Lors de la saisie d'une recette : "));
Calcultot (9,Tot);
Affligne ("- il ne confirme pas le poste budgétaire qu'il a choisi",
         19,9,8,Tot);
Calcultot (11,Tot);
Affligne ("- il ne confirme pas le moment qu'il a choisi",
          20,11,10,Tot);
Calcultot (12,Tot);
Affligne ("- il demande d'effacer la dernière opération",
          21,12,13,Tot);
Affligne ("- il désire quitter la saisie",22,12,14,Tot);
Calcultot (14,Tot);
Affligne ("- il ne quitte pas aprés l'avoir demandé",23,14,12,Tot);
Affligne ("- il quitte réellement la saisie",24,14,7,Tot);
Calcultot (15,Tot);
Affligne ("- il ne confirme pas le montant après avoir cliqué sur ok",
             25,15,12,Tot);
Saisieok (FALSE);
wClrScr(Win2^);
wMove(Win2^,20,2);
PutStr(Win2^,ADR("TRANSITIONS CRITIQUES (suite)"));
wMove(Win2^,20,3);
PutStr(Win2^,ADR("----------------------------"));
wMove(Win2^,1,6);
PutStr(Win2^,ADR("Lors de la saisie d'une dépense : "));
Calcultot (17,Tot);
Affligne ("- il ne confirme pas le poste budgétaire qu'il a choisi",
         8,17,16,Tot);
Calcultot (19,Tot);
Affligne ("- il ne confirme pas le moment qu'il a choisi",
          9,19,18,Tot);
Calcultot (20,Tot);
Affligne ("- il demande d'effacer la dernière opération",
          10,20,21,Tot);
Affligne ("- il désire quitter la saisie",11,20,22,Tot);
Calcultot (22,Tot);
Affligne ("- il ne quitte pas aprés l'avoir demandé",12,22,20,Tot);
Affligne ("- il quitte réellement la saisie",13,22,7,Tot);
Calcultot (23,Tot);
Affligne ("- il ne confirme pas le montant après avoir cliqué sur ok",
             14,23,20,Tot);
wMove(Win2^,1,16);
PutStr(Win2^,ADR("Lors de la correction : "));
Calcultot (7,Tot);
IF Mattrans [7,25] # 0
THEN  Affligne ("- il demande de corriger",18,7,25,Tot);
ELSE  Affligne ("- il demande de corriger",18,7,26,Tot);
END;
Calcultot (26,Tot);
Affligne ("- il demande d'effacer la dernière opération",19,26,27,Tot);
Affligne ("- il désire recommencer la saisie",20,26,28,Tot);
Calcultot (29,Tot);
Affligne ("- il ne confirme pas le montant aprés avoir cliqué sur ok",
           21,29,26,Tot);
wMove(Win2^,1,23);
PutStr(Win2^,ADR("Lors de la fin du programme : "));
Calcultot (30,Tot);
Affligne ("- il ne confirme pas sa demande de fin de session",
          25,30,7,Tot);
Calcultot (31,Tot);
Affligne ("- il revient au menu après présentation de la balance",
```

```
                    26,31,7,Tot);
        imprime := FALSE;
        Saisieok(TRUE);
        IF (imprime = TRUE)
        THEN Impcritiques;
        END;
        DeleteConsole(Win2^);

           END;
           CloseWindow(Win2^);
      END;
      CloseScreen(Scr^);
   END;
END Ecrancritiques;

END Critiques.
```

```
IMPLEMENTATION MODULE Parametres;

FROM ASCII IMPORT CharIsControl;
FROM AmigaDOS IMPORT DateStampRecord,DateStamp;
FROM SYSTEM IMPORT SHORT,BYTE, ADR;
FROM Conversions IMPORT ConvNumberToString;
FROM Storage IMPORT ALLOCATE, DEALLOCATE;
FROM AmigaDOS IMPORT IoErr,CreateDir,FileLock;
FROM AmigaDOSProcess IMPORT Delay;
FROM Intuition IMPORT Window, WindowPtr, ScreenPtr, MenuPtr, GadgetPtr,
  Gadget, WindowFlags, WindowFlagsSet, IDCMPFlags, IDCMPFlagsSet,
  IntuiMessagePtr, ClearMenuStrip, SetMenuStrip, CloseWindow, CloseScreen,
  NewWindow, Image, CustomScreen, OpenWindow, DrawImage, ActivateWindow,
  Requester, InitRequester, Request, RequesterFlagsSet, EndRequest, EndGadget,
  RelVerify,  GadgetFlagsSet, GadgetActivationSet, GadgetMutualExcludeSet,
  OnGadget, OffGadget,ScreenToFront;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, FreeGadgetList,
  AddGadgetTextButton, AddGadgetInteger, AddGadgetString, AddGadgetProp,
  AddGadgetImageButton,GadgetBorder, GadgetTypeReq, GlobalGadgetOpt,
  GadgetOpt;
FROM SimpleIDCMP IMPORT MsgData, WindowProc, ProcIMsg;
FROM SimpleMenus IMPORT BeginMenuStrip, EndMenuStrip, FreeMenuStrip,
  AddMenu, AddMenuItem;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleConsole IMPORT CreateConsole, DeleteConsole, PutStr, GetStr,PutCh;
FROM SimpleConsoleCmds IMPORT wMove, wClrScr, wSetColor, wSetCursor,wClrEndLine
FROM InOut IMPORT WriteString,WriteInt,Write,WriteLn,WriteWrd,OpenOutputFile,
     OpenInputFile,CloseOutput,CloseInput,Done,ReadInt,ReadString,termCH,Read;
FROM Intuition IMPORT StringInfoPtr;
FROM Strings IMPORT  CopyString, CompareString, Relation, ConcatString,
     StringLength, ConvStringToUpperCase,OverwriteWithSubString;
FROM VariablesGlobales IMPORT Mattrans,Str3,Str9,Str20,Str40,TableauParametres,
                              nomuser,Numseance;
FROM Views IMPORT ViewPort, LoadRGB4;
FROM Rasters IMPORT SetRast,RastPortPtr;
FROM Drawing IMPORT SetAPen,Move,Draw,WritePixel;
FROM RealInOut IMPORT WriteReal,ReadReal;
FROM RealConversions IMPORT ConvRealToString,RealToStringFormat,
     ConvStringToReal ;
FROM MathLib0 IMPORT entier,real;

CONST
  WIDCMP = IDCMPFlagsSet {GadgetUp};
  WFlags = WindowFlagsSet {Activate};
  WFlags2 = WindowFlagsSet {Borderless};
  Blanc15 = "               ";
TYPE
  Dernier = RECORD
              Type : Str3;
              X    : CARDINAL;
              Y    : CARDINAL;
            END;

VAR
  Der : Dernier;
  Scr : ScreenPtr;
  Nw : NewWindow;
  Win,Win1,Win2,Win3,Win4 : WindowPtr;
  Indice,I, i : CARDINAL;
  cmap : ARRAY [0..31] OF CARDINAL;
  Ms : MenuPtr;
  G1,G1OK,G12,G13,G14,G15 : GadgetPtr;
  Wp,Wp2,WpOK,Wp3,Wp4,WpImp : WindowProc;
```

```
  Sig : SignalSet;
  ReqOK,Req, Req2,Req3,Req5 : Requester;
  Msg,MsgOK,Msg2,Msg3,Msg4,Msg5 : IntuiMessagePtr;
  Done1,Done2,Done3,Done4,Efface,DoneOK,OK,Conf,UneSemaine,OKNombre,FenWin,
        PremPassage,Done5,Imp2 : BOOLEAN;
  Int : Gadget;
  Confirmation : Str40;
  Seance,Nom2,Max,NomInt,MaxInt : Str20;
  MAX : REAL;
  FileName : Str40;
  Fl : FileLock;
  le : CARDINAL;
  st : Str20;
  For : RealToStringFormat;
  len : INTEGER;
  lene : REAL;
  passe : Str20;
  reconnu : BOOLEAN;
  DV,essai : INTEGER;
  DoneNom : BOOLEAN;
  DernVers : REAL;
  LigneBlanc : ARRAY [0..77] OF CHAR;
  LigneSouligne : ARRAY [0..77] OF CHAR;
  imprime : BOOLEAN;
  Tableau : ARRAY [0..20000] OF CHAR;
  ca : CHAR;
  longfich : INTEGER;
  caract : CHAR;

PROCEDURE JOUR (VAR StrDate:Str40);

VAR

  JJourStr,ANStr : Str20;
  DSR : DateStampRecord;
  MOIS,MOISPREC,Y : INTEGER;
  TrouveMois,TrouveAn : BOOLEAN;
  NBJours : ARRAY [0..11] OF CARDINAL;
  NomMois : ARRAY [0..11] OF Str9;
  Re,Re2,an : REAL;
  X : CARDINAL;
  Quatre : REAL;
  En : INTEGER;
  JJour,AN : INTEGER;

BEGIN

   DateStamp (DSR);
   X := SHORT (DSR.dsDays);
   Y := X;
   X := X + 1;
   AN := 78;
   TrouveAn := FALSE;
   NBJours [0] := 31;
   NBJours [2] := 31;
   NBJours [3] := 30;
   NBJours [4] := 31;
   NBJours [5] := 30;
   NBJours [6] := 31;
   NBJours [7] := 31;
   NBJours [8] := 30;
   NBJours [9] := 31;
   NBJours [10] := 30;
   NBJours [11] := 31;
   NomMois [0] := "Janvier";
   NomMois [1] := "Février";
```

```
   NomMois [2] := "Mars";
   NomMois [3] := "Avril";
   NomMois [4] := "Mai";
   NomMois [5] := "Juin";
   NomMois [6] := "Juillet";
   NomMois [7] := "Aout";
   NomMois [8] := "Septembre";
   NomMois [9] := "Octobre";
   NomMois [10] := "Novembre";
   NomMois [11] := "Décembre";
   WHILE NOT TrouveAn DO
      an := real (AN);
      Quatre := 4.0;
      Re := an/Quatre;
      En := entier (Re);
      Re2 := real (En);
      IF Re=Re2
      THEN
         IF X>366
         THEN
            X := X - 366;
            AN := AN + 1;
         ELSE
            NBJours [1] := 29;
            TrouveAn := TRUE;
         END;
      ELSE
         IF X>365
         THEN
            X := X-365;
            AN := AN+1;
         ELSE
            NBJours [1]:=28;
            TrouveAn := TRUE;
         END;
      END;
   END;
   TrouveMois := FALSE;
   MOISPREC := 11;
   MOIS := 0;
   WHILE NOT TrouveMois DO
      IF  X > NBJours [MOIS]
      THEN X := X - NBJours [MOIS];
           MOISPREC := MOIS;
           MOIS := MOIS +1;
      ELSE TrouveMois := TRUE;
      END;
   END;
   JJour := X;
   CopyString (StrDate,"");
   ConvNumberToString (JJourStr,LONGINT(JJour),FALSE,10,2," ");
   ConcatString (StrDate,JJourStr);
   ConcatString (StrDate," ");
   ConcatString (StrDate,NomMois [MOIS]);
   ConcatString (StrDate," ");
   ConvNumberToString (ANStr,LONGINT(AN),FALSE,10,2," ");
   ConcatString (StrDate,ANStr);

END JOUR;

PROCEDURE GadgetHandler4 (VAR W:Window; VAR Msg4 : MsgData; VAR Gad : Gadget);

BEGIN

   Done4 := TRUE;
```

```
   IF Gad.GadgetID = 1
   THEN imprime := TRUE;
   END;

END GadgetHandler4;


PROCEDURE Saisieok (impr : BOOLEAN);

BEGIN
      BeginGadgetList ();
      AddGadgetTextButton (1,1,ADR(" OK "));
      IF impr = TRUE
      THEN  AddGadgetTextButton (110,1,ADR(" IMPRESSION "));
      END;
      G14 := EndGadgetList ();
      Win3 := CreateWindow (312,226,320,12,WIDCMP,WFlags2,G14,Scr,NIL);
      Done4 := FALSE;
      WHILE (NOT Done4) DO
          Sig := Wait(SignalSet{CARDINAL(Win3^.UserPort^.mpSigBit)});
          LOOP
            Msg4 := GetMsg(Win3^.UserPort^);
            IF (Msg4 = NIL) THEN EXIT; END;
            ProcIMsg (Wp4, Msg4);
          END;
      END;
      CloseWindow(Win3^);
      FreeGadgetList (G14^);

END Saisieok;

PROCEDURE GadgetHandlerImp (VAR w:Window; VAR Msg5 : MsgData; VAR Gad: Gadget);

BEGIN

  CASE Gad.GadgetID OF

    0 : |
    1 : |
    2 : Done5 := TRUE; Imp2 := TRUE |
    3 : Done5 := TRUE;
  END;

END GadgetHandlerImp;


PROCEDURE ImpParam ;

VAR StTemps : Str40;
    J : INTEGER;
    I : INTEGER;

BEGIN

    Imp2 := FALSE;
    WITH WpImp DO
      procGadgetUp := GadgetHandlerImp;
    END;
    BeginGadgetList();
    GadgetTypeReq := TRUE;
    GlobalGadgetOpt (GadgetFlagsSet{},GadgetActivationSet{EndGadget,
                     RelVerify});
    AddGadgetTextButton (60,10,ADR(" L'imprimante est allumée "));
    GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
    GadgetMutualExcludeSet {});
    AddGadgetTextButton (110,30,ADR(" et 'ON-LINE' ? "));
```

```
 GadgetOpt (GadgetFlagsSet {},GadgetActivationSet {RelVerify},
 GadgetMutualExcludeSet {});
 AddGadgetTextButton (110,50,ADR(" OUI "));
 AddGadgetTextButton (195,50,ADR(" NON "));
 G15 := EndGadgetList();
 InitRequester (Req5);
 WITH Req5 DO
         OlderRequest := NIL;
         LeftEdge := 150;
         TopEdge := 140;
         Width := 340;
         Height := 80;
         ReqGadget := G15;
         ReqBorder := NIL;
         ReqText := NIL;
         Flags := RequesterFlagsSet {};
         BackFill := BYTE(8);
         ReqLayer := NIL;
         ImageBMap := NIL;
 END;


 IF Request (Req5,Win2^)
 THEN
 Done5 := FALSE;
 WHILE (NOT Done5) DO
       Sig := Wait(SignalSet{CARDINAL(Win2^.UserPort^.mpSigBit)});
       LOOP
         Msg5 := GetMsg(Win2^.UserPort^);
         IF (Msg5 = NIL) THEN EXIT; END;
         ProcIMsg (WpImp, Msg5);
       END;
 END;
 END;
IF Imp2 = TRUE
THEN

 OpenOutputFile ("prt:");
 IF Done
 THEN
    WriteLn;
    WriteLn;
    WriteLn;
    WriteString("Nom de l'utilisateur : ");
    WriteString(nomuser);
    WriteLn;
    WriteLn;
    WriteString("Numéro de la séance : ");
    WriteString(Numseance);
    WriteLn;
    WriteLn;
    StTemps :="";
    JOUR ( StTemps);
    WriteString("Date de l'impression : ");
    WriteString(StTemps);
    WriteLn;
    WriteLn;
    WriteLn;
    WriteLn;
    WriteString("                                TRACES D'UTILISATION");
    WriteLn;
    WriteString("                                --------------------");
    WriteLn;
    WriteLn;
    CloseOutput ();
    I := 0;
```

```
      J := 0;
      OpenOutputFile ("prt:");
      WHILE (I < longfich) DO
                  IF (J > 110) THEN
                       CloseOutput ();OpenOutputFile("prt:");J:=0;
                  END;
                  Write (Tableau[I]);
                  I := I + 1;
                  J := J + 1;
       END;
       WriteLn;
       WriteLn;
       WriteLn;
       CloseOutput ();
   END;
   END;
   FreeGadgetList(G15^);
END ImpParam;



PROCEDURE Paramet (VAR DernVers : REAL);

VAR I,J,Long,Valact,Valprec : INTEGER;
    St : Str20;
    Filechaine :Str40;
    ent : INTEGER;
    ree,ree2 : REAL;
    OKSeance : BOOLEAN;

BEGIN

    WITH Wp4 DO
      procGadgetUp := GadgetHandler4;
    END;
    Scr := CreateScreen (640,240,4,NIL);
    IF (Scr # NIL) THEN
      cmap[00] := 0000H;
      cmap[01] := 0FFFH;
      cmap[02] := 000FH;
      cmap[03] := 03F1H;
      cmap[04] := 0FE0H;
      cmap[05] := 0F0CH;
      cmap[06] := 00FFH;
      cmap[07] := 0870H;
      cmap[08] := 0E00H;
      cmap[09] := 0F90H;
      cmap[10] := 098FH;
      cmap[11] := 007CH;
      cmap[12] := 000FH;
      cmap[13] := 070FH;
      cmap[14] := 0C0EH;
      cmap[15] := 0C08H;
      cmap[16] := 0620H;
      cmap[17] := 0E52H;
      cmap[18] := 0A52H;
      cmap[19] := 0FCAH;
      cmap[20] := 0333H;
      cmap[21] := 0444H;
      cmap[22] := 0555H;
      cmap[23] := 0666H;
      cmap[24] := 0777H;
      cmap[25] := 0888H;
      cmap[26] := 0999H;
      cmap[27] := 0AAAH;
      cmap[28] := 0CCCH;
```

```
cmap[29] := 0DDDH;
cmap[30] := 0EEEH;
cmap[31] := 0FFFH;
LoadRGB4 (Scr^.ViewPort,ADR (cmap),16);
Win2 := CreateWindow (0,0,640,240,WIDCMP,WFlags,NIL,Scr,NIL);
IF (Win2 # NIL)
THEN IF CreateConsole(Win2^)
     THEN
      OKSeance := FALSE;
      WHILE (NOT OKSeance) DO
          wClrScr(Win2^);
          wMove(Win2^,30,2);
          PutStr(Win2^,ADR("TRACES D'UTILISATION"));
          wMove(Win2^,30,3);
          PutStr(Win2^,ADR("--------------------"));
          wMove(Win2^,7,10);
          PutStr(Win2^,ADR("Numéro de la séance [ de 1 à "));
          DernVers := DernVers + 1.;
          ConvRealToString(DernVers,St,0,Decimal);
          DernVers := DernVers - 1.;
          PutStr(Win2^,ADR(St));
          PutStr(Win2^,ADR(" ] : "));
          wSetCursor(Win2^,TRUE);
          GetStr(Win2^,ADR(Numseance),SIZE(Numseance));
          wSetCursor(Win2^,FALSE);
          ree := ConvStringToReal (Numseance);
          ree := ree - 1.;
          ent := entier (ree);
          ree2 := real(ent);
          IF (ree2 = ree) AND (ent >= 0) AND (ree <= DernVers )
          THEN OKSeance := TRUE;
               ConvRealToString(ree,Seance,0,Decimal);
          END;
      END;
      wClrScr(Win2^);
      Filechaine := "Param:";
      ConcatString(Filechaine,nomuser);
      ConcatString(Filechaine,Seance);
      ConcatString(Filechaine,".TRACE");
      OpenInputFile(Filechaine);
      ScreenToFront(Scr^);
      IF Done
      THEN
         Read (caract);
         WHILE (NOT CharIsControl(caract)) DO
           Read (caract);
         END;
         I := 0;
         longfich := 0;
         WHILE (Done) DO
               Read(Tableau[I]);
               I := I + 1;
               longfich := longfich + 1;
         END;
         CloseInput;

         I := 0;
         J := 0;
         Win4 := CreateWindow (150,22,340,175,WIDCMP,WFlags,NIL,Scr,NIL)
         IF (Win4 # NIL)
         THEN
           IF CreateConsole(Win4^)
           THEN
            wClrScr(Win4^);
            wSetCursor(Win4^,FALSE);
            wMove(Win4^,1,1);
```

```
                WHILE (I < longfich ) DO
                    IF CharIsControl (Tableau[I])
                    THEN J := J + 1;
                    END;
                    IF (J > 16)
                    THEN Saisieok(FALSE);
                         J :=0;
                    END;
                    PutCh(Win4^,Tableau[I]);
                    I := I + 1;
                END;
                imprime := FALSE;
                Saisieok(TRUE);
                DeleteConsole(Win4^);
              END;
              CloseWindow(Win4^);
            END;

            IF imprime = TRUE
            THEN ImpParam;


            END;
         END;
         DeleteConsole(Win2^);
       END;
       CloseWindow(Win2^);
  END;
  CloseScreen(Scr^);
END;
END Paramet;

END Parametres.
```

```
DEFINITION MODULE Critiques;
FROM VariablesGlobales IMPORT Str20,Mattrans;
PROCEDURE Ecrancritiques;
END Critiques.

DEFINITION MODULE Parametres;
FROM VariablesGlobales IMPORT Str20,Mattrans;
PROCEDURE Paramet (VAR DernVers: REAL);
END Parametres.
```

Réalisation, impression et reliures : SPRL Centre de Reprographie - Tél. : 081/22.87.72